uc3m | Universidad Carlos III de Madrid

Grado Ingeniería de Sistemas de Comunicaciones
Curso académico: 2016-2017

*Trabajo Fin de Grado*

# "CUENTIX: Desarrollo de una aplicación avanzada de cuentacuentos para dispositivos móviles"

---

Adrián Iriberri Bercial

Tutor: David Griol Barres

Universidad Carlos III Madrid | Leganés, septiembre 2017

# AGRADECIMIENTOS

Durante estos años de carrera ha habido muchas personas que han logrado, indirectamente, la realización de la carrera.

Primeramente, quisiera dar las gracias a mi tutor David Griol. Ha conseguido, con su optimismo y confianza, que aumentaran mis ganas de seguir aprendiendo y valorando cada paso en la realización del proyecto.

Agradecer a todos los profesores de la Universidad Carlos III de Madrid su ayuda. Agradecer también a todos los compañeros que vinieron para quedarse en mi vida, que considero no sólo amigos, sino también parte de mi familia.

Gracias a mi familia y amigos, por su apoyo y compartir conmigo mis ganas de realizar el proyecto.

Gracias a Óscar Gallo y a Gabriel Castellanos, sin los cuales este proyecto no podría haberse visto realizado.

Gracias a mi pareja, Hayati, su paciencia y su apoyo durante estos años han sido, en muchas ocasiones, el oxígeno que necesitaba para poder seguir.

Por último, dar las gracias a mis padres, los cuales jamás han dudado de mis capacidades y siempre han estado ahí para cualquier cosa que necesitase. Sin vuestra ayuda no podría haber llegado a ser la persona que ahora mismo me enorgullece decir que soy.

Muchas gracias a todos los que habéis aportado tanto en mí como en mis logros ese granito de arena que tanto he necesitado y tanto he agradecido. Gracias.

# RESUMEN

El proyecto que se menciona en este documento trata sobre una aplicación llamada Cuentix y desarrollada con la herramienta Android Studio, enfocada en todo momento a su fácil uso por diferentes perfiles de usuario, en especial por los más jóvenes.

El trabajo trata sobre un cuenta-cuentos interactivo en el que el usuario podrá interactuar con la aplicación mediante reconocimiento de voz y el motor de Google de síntesis de voz. Podrá tanto leer un cuento como editarlo o, incluso, crear uno desde cero con su propia historia y sus propias imágenes.

En este documento se hará una breve exposición sobre el mundo de las tecnologías en la actualidad, las características del Sistema Operativo al que está destinado y el lenguaje de programación en el que se ha realizado. También se realizará una explicación detallada sobre la aplicación, no sólo a nivel de funcionamiento de cara al usuario, sino también del aspecto interno más orientado al desarrollador. Por último, se hará mención a la gestión del mismo y las posibilidades que ofrece su implementación.

Para finalizar, indicar que el motivo de este trabajo no es solamente el de entretener al público, sino también de poder ser usado para tratar distintas enfermedades degenerativas como el Alzheimer.

**Palabras clave:** Cuentix, cuento, reconocimiento de voz, motor TTS, Android, Android Studio.

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# CAPÍTULO 1 – INTRODUCCIÓN

En las siguientes líneas se expondrá brevemente una vista global sobre el mundo de las tecnologías, más concretamente de la rama que ocupa este proyecto, los objetivos que se pretenden alcanzar, el desarrollo de todo el proyecto, los recursos utilizados para realizarlo y cuáles son los contenidos ofrecidos en esta memoria.

## 1.1. EL SER HUMANO Y LA NECESIDAD DE ENSEÑAR

El ser humano, desde el origen de su existencia ha dejado su marca en la historia, recordando y enseñando sus costumbres y vivencias. Ese afán ha ido prevaleciendo durante los años, aumentando nuestra capacidad para hacer llegar a cualquier persona, de cualquier lugar y de cualquier época estos conocimientos.

Al igual que ha ido evolucionando el intelecto del individuo, también ha ido evolucionando la forma de externalizar la información. Y en esta era, la era tecnológica, cada vez es más fácil poder transmitirla.

Desde pequeños, la capacidad de aprendizaje ha ido adaptándose a la época en la que se encontrase la persona y, debido a ello, ha sido necesario cambiar y adaptar su modelo.

## 1.2. MIRADA AL MUNDO DE LAS TECNOLOGÍAS

El mundo tecnológico ha sufrido un desarrollo tan elevado y tan rápido que ha conseguido integrarse no sólo en las áreas de conocimiento y producción, sino en la mentalidad de la población que ha hecho de ella un elemento tan sumamente imprescindible que nadie sería capaz, a día de hoy, de poder imaginarse el mundo sin ella.

Con la llegada del ordenador (o computador, como se denominaba en aquel entonces) por el alemán Honrad Zuse, se inició un periodo de bonanza para las nuevas tecnologías. Si bien es cierto que las primeras computadoras eran simples aparatos con escasas instrucciones que hacían el deleite de escasos afortunados. Pero no fue hasta la aplicación de los circuitos integrados cuando este producto se llevó a un nuevo nivel, el del pequeño consumidor y, con él, la increíble velocidad de su desarrollo y evolución.

Más tarde, con la reconocida conclusión satisfactoria de Arpanet, se abrió camino a la creación de lo que hoy conocemos como Internet (como medio de transmisión), lo que dio lugar al "Boom tecnológico" al conseguir que los dos conceptos "Internet" y "telefonía móvil" consiguieran acoplarse uno junto al otro.

Debido al gran impacto social que tuvo en la sociedad, la búsqueda de la excelencia llevó a una mejora de los productos y a la cercanía de los mismos a todas las clases sociales. Pero, esta tecnología seguía siendo sólo utilizable a un reducido sector de la sociedad por lo que, gracias a este cambio, se dio un paso de gigante con la invención del teléfono móvil.

## 1.3. OBJETIVOS

Como he mencionado anteriormente, el mundo de las tecnologías es un mundo en constante cambio y actualización; lo que hoy puede ser un "bombazo" tecnológico, mañana puede estar obsoleto debido a otra aplicación mejor y actualizada.

Dado que este proyecto no es sino una actualización de los famosos libros "cuenta tu historia", un objetivo importante fue el de hacerlo simple pero a la vez interesante para el grupo al que va dirigido, los niños.

Pero, para la realización de este objetivo, eran necesarios unos claros y amplios conocimientos sobre la realización del mismo, en este caso, mediante programación Android. Fue una ardua tarea conseguir estos conocimientos, pero una vez conseguidos mediante libros y "cursos" en Internet, el "camino" ya tenía un punto de partida y otro de final.

A partir de este punto, lo más importante era el camino, ¿cómo se debía realizar la aplicación? ¿Quería que fuera dinámica, que hablara al usuario, que pudiéramos hablar a la aplicación o sólo pudiéramos interactuar de forma táctil?

Tras mucho tiempo debatiendo conmigo mismo y tras muchas indecisiones, conseguí decidir cómo quería que fuera la aplicación, con los siguientes puntos:

- Debía ser una aplicación que no aburriera a los usuarios.

- Debía ser una aplicación que se pudiera usar tanto por la voz como táctilmente (todos sabemos que a los pequeños les gusta mucho tocar las pantallas de los dispositivos).

- Debía ser lo más sencilla posible tanto para leer un cuento como para crearlo.

Una vez resuelto el problema, sólo quedó unir todos los puntos, escribir los comandos correctos y rellenar los huecos que me iba encontrando. Como resultado, se creó “Cuentix”.

Teniendo todo esto en cuenta, y como breve resumen, el objetivo principal de este proyecto no es sólo el de “digitalizar” un cuento, sino poder abrir paso a otras nuevas aplicaciones y actualizaciones que sean capaces de mejorar lo ya creado e, incluso, dar ideas para otros posibles proyectos más ambiciosos.

## 1.4. DESARROLLO DE LA APP

Tras mucho debatir tanto conmigo mismo como con mi familia y gracias a experiencias no tan buenas con proyectos anteriores, mi pensamiento hacia la realización del TFG se orientó a un proyecto en el que primaran aspectos como que fuera útil para cualquier persona, que pudiera ser usado por todo el mundo, que fuera gratuito y, sobretodo, que diera pie a posibles mejoras e incluso, nuevas ideas.

Las siguientes líneas definirán el camino que tuve que recorrer desde el primer momento en el que decidí qué tipo de proyecto quería hacer hasta la redacción de estas líneas y su futura presentación, pasando por aspectos tan básicos pero importantes como qué herramientas usar, qué funciones debe tener la aplicación o, incluso, algo tan sencillo como la cara de un niño mientras usa la app.

Mediante la figura 1, se tendrá una idea más clara de los puntos que se divide el camino mencionado.

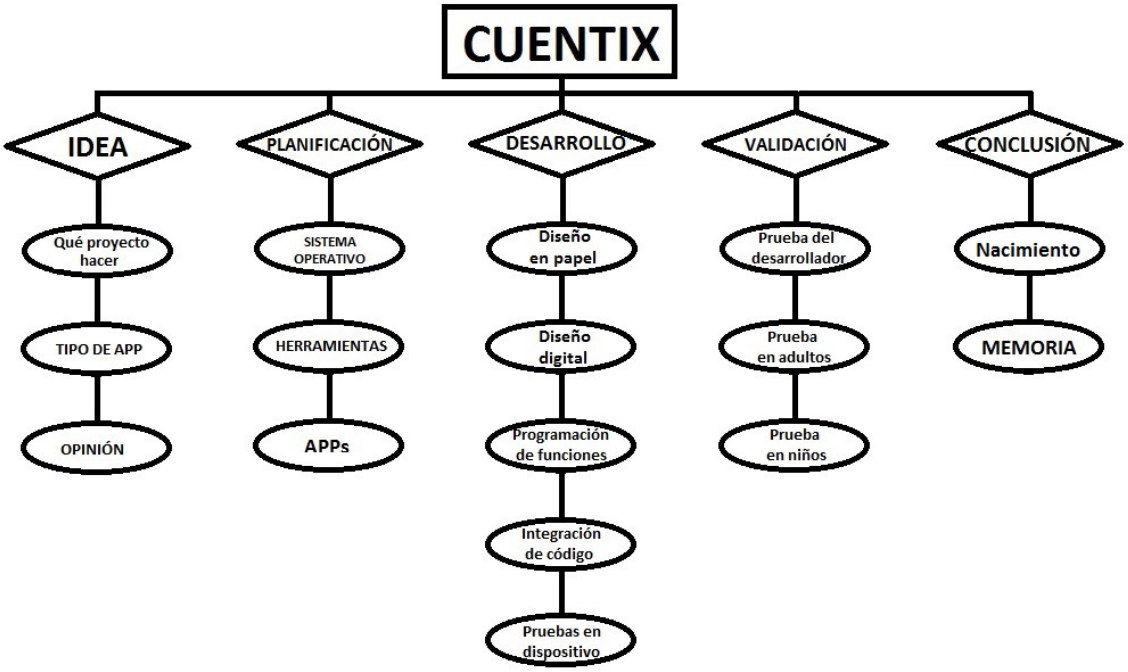



*Figura 1 - Desarrollo de Cuentix*

### 1.4.1. IDEA

**- Qué proyecto hacer:** Es probablemente uno de los grandes y principales problemas para cualquier persona que quiera realizar el TFG. Es un aspecto importante saber qué tipo de proyecto se quiere hacer, sobretodo porque de su elección dependerá, en mayor medida, de su finalización. Siempre me fascinó el mundo de la programación y, tras un contacto fallido con un primer proyecto, tenía claro el tipo de proyecto que quería realizar esta vez. Un proyecto sobre programación de una app.

**- Tipo de App:** Una vez resuelto el problema de qué proyecto quería realizar, ahora se me cruzaba un problema en el camino: ¿qué aplicación quería hacer? Y es que este aspecto me hizo recordar a unas palabras de mi padre "¿Y ahora qué? Si ya está todo inventado". Toda la razón cada vez que dice esa frase pero, ¿Y si no todo está inventado. Y si pudiera coger algo de mi infancia y poder actualizarlo a los tiempos que corren? Así comencé a encauzar la idea del tipo de aplicación que quería desarrollar.

**- Opinión:** Estaba llegando a la puerta que iba a iniciar el camino de mi TFG; sin embargo, no encontraba esa idea clave que me resolviera todas mis dudas, y fue mi sobrino de 3 años quien me ayudó con la solución. Un acto tan simple como pedirme que le leyera un cuento fue el detonante, fue la corriente que hiciera que se encendiera la bombilla: ¿Y por qué no hacer un cuentacuentos, pero que pudieras crear tus propios cuentos y, a la vez, fuese interactivo?

### 1.4.2. PLANIFICACIÓN

**- Sistema operativo:** Tras la elección del proyecto, un cuentacuentos interactivo, debía decidir en dónde se debería visualizar. Este aspecto fue muy sencillo, en un móvil o en una Tablet. Pero hay muchos SO para muchas tablets y móviles, por lo que decidí que si quería que fuera gratuita, debería estar implementada en Android, ya que tiene muchísimas aplicaciones de coste cero y el lenguaje es bien conocido (muchos años de universidad programando en C y en Java).

**- Herramientas:** Al igual que hay muchos SO en los que poder programar, también hay muchas opciones para programar en Android; las dos más conocidas son Eclipse y Android Studio. Tras ver sus diferencias lo tuve claro: ¿por qué usar una herramienta que emula Android, cuando puedo usar una aplicación propia del desarrollador del Sistema Operativo? Así me decidí por realizarla en Android Studio.

**- APPs parecidas:** El último punto por decidir de la planificación de la app era tener una posible idea de cómo realizarla. Sabía que iba a ser un cuento, en el que la aplicación pudiera emitir sonido como un estilo narrador y que el usuario pudiera "mandarle" la acción al dispositivo vía oral o bien, táctilmente. Sólo hizo falta una pequeña búsqueda en "Google Play" para tener una idea de todos estos puntos.

### 1.4.3. DESARROLLO

**- Diseño en papel:** Ya tenía claro qué tipo de proyecto quería hacer, que tipo de aplicación y en qué lenguaje; ahora sólo me faltaba decidir cómo iba a ser la aplicación tanto interna como externamente. Gracias a los consejos de David Griol, y con un poco de imaginación, conseguimos tener un diseño previo en papel de cómo podía ser y qué era necesario para llevarla a cabo.

**- Diseño digital:** Gracias a ese diseño en papel, traspasarlo a un programa digital se hizo realmente sencillo. Pero fue en este punto en el que se empezó a vislumbrar cómo sería la interfaz de la aplicación, es decir, su cara visible al público.

**- Programación de funciones:** Era hora de empezar a "programar" en sí. Uniendo los conocimientos adquiridos en la universidad, un poco de la ayuda de cursos en la web y las pautas aportadas por David Griol, fui implementando las distintas funciones necesarias.

**- Integración del código:** Tras la implementación de las funciones, era la hora de ir incorporándolas una a una a la estructura final e ir completando los puntos que se debían realizar una vez las funciones se integraran.

**- Pruebas en dispositivo:** Este punto va de la mano con el punto anterior, ya que fue necesario un gran número de pruebas y cambios en el código para que todo "pegara" de la forma correcta y pudiera darse por "previamente" finalizada la aplicación.

### 1.4.4. VALIDACIÓN

**- Prueba del desarrollador:** Una vez finalizada la aplicación y comprobado que podía "correr" en el dispositivo, fue el momento de realizar las pruebas de funcionamiento. Fue en este momento en el que se pudo ver la aplicación ejecutándose con normalidad, generando buenas sensaciones. ¿Opinaría lo mismo el resto de personas?

**- Prueba en adultos:** Era necesario que un adulto sin experiencia en programación usara por primera vez la aplicación, en este caso quien seleccioné como "conejillo de indias" fue a mi padre. Tras unas breves explicaciones y alguna que otra desesperación, concluyó su propia validación de forma satisfactoria.

**- Prueba en niños:** Era el momento decisivo en el que un usuario al que realmente iba dirigido el proyecto realizara la evaluación. Y quien mejor que con el que empezó todo este "lío", mi sobrino. Tras unas más que elevadas explicaciones y con ayuda de algún miembro más de la familia, comenzó a usar la aplicación y así siguió durante unas cuantas horas.

### 1.4.5. CONCLUSIÓN

**- Nacimiento:** Tras la implementación y las pruebas, ya se podía determinar que la aplicación había nacido. Era momento de explicar todo el proceso de realización en una memoria, en esta memoria.

**- Memoria:** La memoria debía ser clara, cercana al lector y completa, pero a la vez fácil de leer. Mentiría si en estas líneas dijera que ha sido la parte más "bonita" del TFG; sin embargo, poder explicar las ideas básicas, funcionamiento y finalidad de la aplicación compensó con creces la pesadumbre de escribir.

## 1.5. ESTRUCTURA DE LA MEMORIA

En las siguientes líneas se expondrán los capítulos que conforman la memoria, además de una breve descripción de cada uno de ellos.

**1. Introducción:** Incluye un breve resumen del desarrollo de la tecnología en las últimas décadas, el factor humano en este desarrollo, los objetivos que se persiguen con el proyecto, el proceso llevado a cabo desde la búsqueda del TFG hasta su finalización, la estructura de la memoria y las herramientas empleadas para la consecución de la app.

**2. Estado del arte:** En este punto se definirá lo que es un dispositivo móvil, sus tipos y los Sistemas Operativos presentes en la actualidad, además se darán unas referencias al SO Android y sus versiones más actuales. Por último, se describirán los sistemas de diálogo, la conversión de texto a voz (TTS), reconocimiento de voz y la base de datos utilizada.

**3. Descripción de la App:** Se aportará una imagen visual previa de la aplicación, su estructura general y, en más detalle, cada bloque que compone la aplicación.

**4. Evaluación:** Se harán referencias a pruebas tanto del desarrollador como de usuarios de distintas edades. También se describirán los fallos y discusiones que han podido surgir durante su validación.

**5. Conclusiones y trabajo futuro:** En este punto se aportarán las conclusiones sobre la realización de la aplicación y de la memoria, las posibles ideas para mejorar y/o actualizar la aplicación y cómo ha sido mi experiencia con el proyecto, tanto con la aplicación como con la memoria.

**6. Gestión del proyecto:** En este capítulo se explicarán los requisitos legales que debe cumplir nuestra aplicación antes de poder salir al mercado, los gastos tanto por horas empleadas en la elaboración de la app como de material y su justificación y se dará una visión sobre la aplicación, orientada al ámbito social y económico.

**Summary in English:** Contiene un breve resumen, en inglés, de las partes más importantes de la memoria.

## 1.6. RECURSOS EMPLEADOS

**Para la realización de la aplicación ha sido necesario:**

- Ordenador: HP Intel Core 2 DUO.
- Smartphone: Samsung Galaxy A5.
- Tablet: Huawei Mediapad M2 10.
- Android Studio (programa + paquetes descargados + SDK + AVD).
- Paint y Photoshop para la realización y retoque de imágenes.
- Conexión a Internet.

**Para la realización de la memoria ha sido necesario:**

- Ordenador: HP Intel Core 2 DUO.
- Microsoft Word.
- Conexión a Internet.
- Paint t Photoshop para edición de figuras

# CAPÍTULO 2 – ESTADO DEL ARTE

En este capítulo expondremos una visión general sobre los dispositivos móviles existentes en la actualidad, además de los sistemas operativos (SO) más utilizados para su desarrollo.

Por otra parte, analizaremos el SO utilizado en este proyecto y el programa usado para llevarlo a cabo.

## 2.1. DISPOSITIVOS MÓVILES

Desde la invención del primer Smartphone en 1992 por IBM Simon se ha evolucionado bastante en el diseño y rendimiento de los dispositivos móviles, lo que ha provocado una bajada en su presupuesto y un acercamiento a todos los sectores de la sociedad.

Coloquialmente, un dispositivo móvil (computadora de bolsillo o de mano) se puede definir como un aparato de pequeño tamaño, con algunas capacidades de procesamiento, con conexión permanente a Internet, de memoria limitada y diseñado para una función, pero con capacidad para llevar a cabo otras funciones al mismo tiempo.

Una gran cantidad de dispositivos electrónicos se incluyen dentro de esta definición, desde teléfonos, hasta tablets, incluyendo PDAs y eBooks. Por ello, antes de hacer una clasificación sobre los tipos de dispositivos móviles existentes en el mercado actual, deberemos detallar una serie de características esenciales para poder denominar un dispositivo como “dispositivo móvil”.

- Son aparatos pequeños.
- La mayoría son portables.
- Poseen capacidad de procesamiento.
- Conexión permanente o intermitente a una red de Internet.
- Tienen memoria limitada (RAM, SD, etc.).
- Uso personal e individual de un usuario.

## 2.2. TIPOS DE DISPOSITIVOS MÓVILES

Atendiendo a las características expuestas anteriormente, podemos hacer una pequeña clasificación de los distintos dispositivos móviles existentes.

### 2.2.1. REPRODUCTORES DE MÚSICA

Diseñados especialmente para escuchar y descargar archivos de música.

### 2.2.2. LIBROS ELECTRÓNICOS (EBOOKS)

Equipos que permiten guardar cierta cantidad de libros y documentos para leerlos en cualquier momento y lugar, adecuando el formato y el tamaño del texto a las necesidades del usuario.

### 2.2.3. ORDENADORES PORTÁTILES

Ordenador (dispositivo electrónico capaz de almacenar información y tratarla automáticamente mediante operaciones matemáticas y lógicas controladas por programas informáticos) personal de reducido tamaño y peso con una batería incorporada aportando autonomía limitada.

### 2.2.4. GRABADORES MULTIMEDIA

Dispositivo que posibilita la grabación de datos en formato de audio o video (principalmente). En este apartado tenemos las cámaras digitales de vídeo y fotos.

### 2.2.5. VIDEOCONSOLAS

Aparato electrónico de entretenimiento que ejecuta juegos contenidos en discos magnéticos, ópticos o tarjetas de memoria. Pueden ser utilizados con una pantalla o directamente en el dispositivo (videoconsolas portátiles).

### 2.2.6. SMARTPHONES

También llamado “teléfono inteligente”. Se trata de un teléfono que cuenta con las funciones básicas de un móvil (recibir/realizar llamadas y mensajes, juegos) y con algunas añadidas (conexión a Internet, pantalla táctil, capacidad multimedia, instalación de aplicaciones).

### 2.2.7. TABLETS

Se trata de una computadora portátil que se caracteriza por tener pantalla táctil, más grande que un teléfono inteligente (posee características similares) pero más pequeño que un netbook. Su tamaño ronda entre las 8 y las 13 pulgadas.

## 2.3. SISTEMAS OPERATIVOS (SO) PARA DISPOSITIVOS MÓVILES

Hasta hace poco, los únicos aspectos en los que se debía fijar un consumidor a la hora de obtener un dispositivo era su aspecto y la duración de su batería. Sin embargo, en la actualidad, el sistema operativo usado en dicho dispositivo se ha convertido en una de las características más importantes a la hora de elegirlo.

Un sistema operativo móvil controla un dispositivo móvil, al igual que los ordenadores utilizan sus sistemas operativos para ser gestionados. Son mucho más simples y están orientados especialmente a la conectividad inalámbrica.

La elección de uno u otro SO dependerá de las características ofrecidas por cada uno, siempre dependiendo del tipo de usuario que está escogiendo el aparato.

Los SO más utilizados actualmente por los dispositivos móviles son los siguientes:

### 2.3.1. BlackBerry OS

Sistema desarrollado por BlackBerry en 1999 mediante la creación de las primeras PDAs. Obtuvo gran importancia en el mercado debido a su uso emergente en móviles con acceso a Internet orientados a las empresas.

### 2.3.2. Firefox OS

Basado en HTML5 con núcleo Linux, desarrollado por Mozilla Corporation y anunciado en febrero de 2013. Está enfocado especialmente para los dispositivos móviles de gama baja y diseñado para permitir la comunicación directa con el hardware del dispositivo mediante JavaScript.

Sin embargo, en febrero de 2016, Mozilla anunció que dejaría de ofrecer soporte y actualizaciones para la versión móvil, para centrarse en trasladar dicho sistema operativo a Smart TVs.

### 2.3.3. Windows Phone

Anteriormente llamado Windows Mobile y desarrollado por Microsoft, se basa en el núcleo del sistema operativo Windows CE. Está diseñado para ser similar a las versiones de escritorio de Windows. A pesar de ser el SO más seguro del mercado (imposible acceder al contenido del teléfono mediante una aplicación) y del lanzamiento de la versión 10, sus ventas han caído considerablemente.

### 2.3.4. iOS

Denominado en sus inicios como iPhone OS y desarrollado por Apple, se creó originalmente para iPhone. Actualmente, da vida a todos los dispositivos pertenecientes a dicha compañía. Se caracteriza por su simplificación y optimización, aportándole mayor potencia y fluidez.

### 2.3.5. Android

Sistema operativo número uno en cuota del mercado en los últimos años y basado en el núcleo Linux. Fue diseñado por Android Inc. pero respaldado económicamente por Google (absorbió la empresa en 2005).

La mejor característica de este SO es su carácter abierto. Además, la mayoría de las aplicaciones están escritas en Java y su desarrollo y utilización es totalmente gratuito.

Dado que la aplicación desarrollada para este proyecto se ha desarrollado para la plataforma Android, se van a dedicar el siguiente punto al estudio de dicha plataforma.

## 2.4. PLATAFORMA ANDROID

En este punto se expondrá la historia de Android, su arquitectura y sus características, además de las versiones más actuales hasta la fecha.

### 2.4.1. HISTORIA

La idea de Android se creó bajo el nombre de la compañía Android Inc, una empresa modesta formada por cuatro jóvenes en California en el año 2003. No tenían muchas expectativas sobre la longevidad de dicho proyecto, pero sí tenían claro que sería un SO abierto, basado en Linux y con una característica principal: ofrecer una experiencia libre e innovadora al usuario.

La empresa continuó creciendo poco a poco durante los dos años siguientes hasta que, en 2005, Google decidió comprar la empresa por 50 millones de dólares. El mundo tecnológico se quedó boquiabierto ante esta jugada por parte de la empresa americana, no sólo por el pequeño tamaño que comprendía Android Inc, sino también por las razones ante dicha acción (simplemente añadió que se trataba de una buena idea con gran futuro).

Gracias al amparo de una empresa como Google, y la participación de otras empresas pequeñas adquiridas al igual que Android Inc, el SO evolucionó y mejoró a una gran velocidad; sin embargo, el anonimato sobre el producto tanto a la prensa como al resto de los medios de comunicación, posicionaron a Android en un segundo plano.

Los grandes esfuerzos tanto de los desarrolladores como de los fabricantes de hardware y software hicieron posible la creación y, por consiguiente, el anuncio de la primera versión de Android en noviembre de 2007: Android 1.0 Apple Pie.

Actualmente, existe una gran comunidad de desarrolladores creando aplicaciones tanto profesionales como amateurs, lo que ha provocado que el número de aplicaciones disponibles en Google Play (plataforma de distribución digital de aplicaciones móviles para los dispositivos con sistema operativo Android) supere el millón.

Las unidades vendidas de teléfonos inteligentes que utilizan Android como sistema operativo se ubican en el primer puesto de todos los países a partir del año 2010, lo que ha incentivado aún más el interés por conocer, aprender y programar este SO por parte de la población.

Gracias a todos estos aspectos, Android ha conseguido ponerse a la cabeza tanto de venta de productos como de creación de aplicaciones en todo el mundo en sólo una década.

### 2.4.2. CARACTERÍSTICAS

Como ya hemos explicado antes, Android es un sistema operativo de código abierto para dispositivos móviles creado por Google (y sus empresas asociadas), basado en Linux y con aplicaciones preinstaladas para facilitar el uso.

Sus características principales y comunes a todos los dispositivos son:

- Permite el reemplazo y la reutilización de los componentes.

- Navegador integrado basado en el motor de código abierto Source Webkit.

- Base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones, denominado SQlite.

- Soporte para medios con formatos comunes de audio, video e imágenes (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).

- Base de llamadas de instancias muy similar a Java adaptada a dispositivos móviles (Máquina virtual Dalvik).

- Telefonía GSM.

- Interfaces de red CDMA, GSM, Bluetooth, EDGE, 3g y Wifi.

- Cámara, GPS, brújula y acelerómetro.

- Pantalla multitáctil.

- Biblioteca de gráficos 2D y 3D OpenGL ES 1.0

- Entorno de desarrollo completo que incluye un emulador de dispositivos, herramientas para la depuración, la memoria y perfiles de rendimiento, y un plug-in para el IDE de Eclipse.

- Uso de Google Play: catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.

- Búsqueda en Google a través de voz está disponible como "Entrada de Búsqueda" desde la versión inicial del sistema.

### 2.4.3. SEGURIDAD

Dado que Android es un sistema de coste cero y de código abierto, hay un elevado número de aplicaciones malintencionadas que intentan violar la privacidad del usuario. Por eso, la seguridad en Android es un aspecto sumamente vital.

Gracias al sistema de permisos, se consigue impedir que las aplicaciones realicen acciones comprometidas, si previamente no se ha solicitado el permiso correspondiente.

Android propone un esquema de seguridad [1] sin imponer un sistema centralizado y controlado por una única empresa. Por esto, Android se fundamenta en tres pilares:

1. Android está basado en Linux, por lo que se aprovechará la seguridad que conlleva este sistema.

2. Toda aplicación debe ser firmada por el autor con un certificado digital.

3. Es necesario un modelo de permisos si queremos que una aplicación acceda a partes del sistema que comprometan la seguridad del dispositivo.

Con todo esto, podremos denotar que el sistema Android cubre, de forma óptima, la seguridad necesaria. Además, debemos destacar que con cada versión, aumenta la seguridad y los permisos de cada aplicación, lo que es aliciente más para optar por este sistema operativo.

## 2.4.4. ARQUITECTURA

La arquitectura de Android está formada por 5 capas que facilitan al desarrollador la creación de aplicaciones, como indica la figura 2.

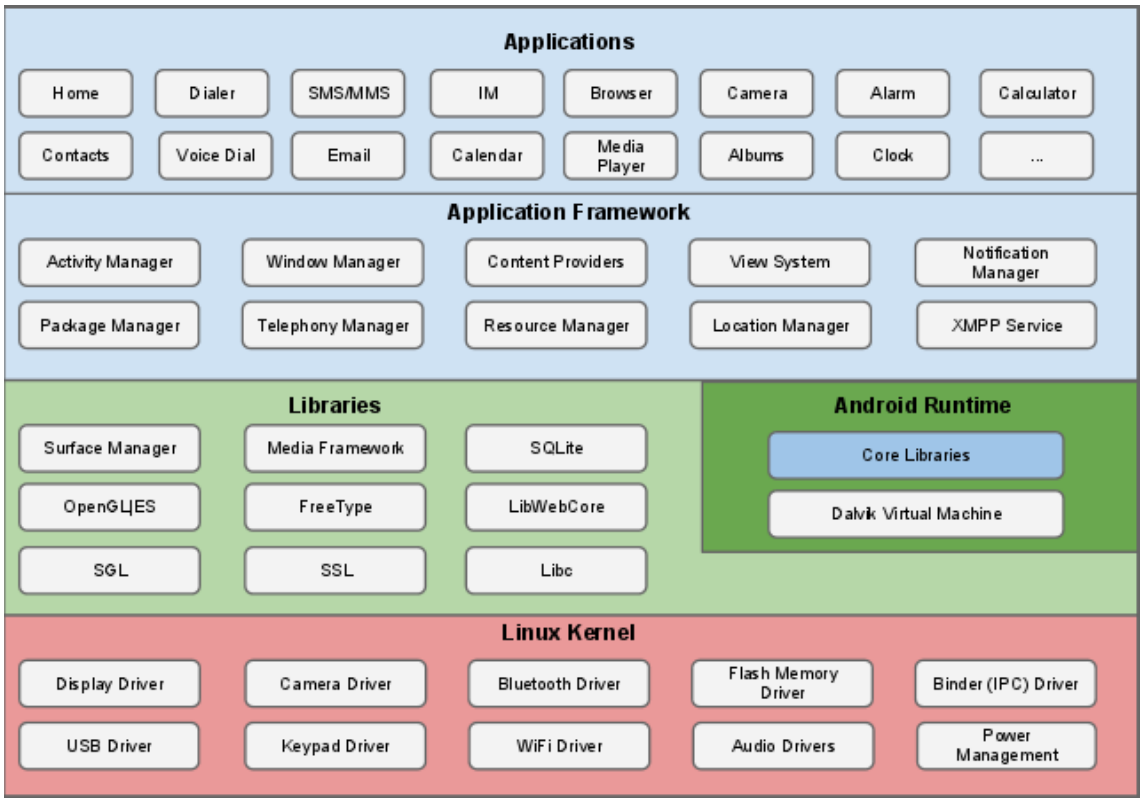



*Figura 2 - Arquitectura de Android*

La distribución expuesta permite acceder a las capas inferiores mediante las librerías, lo que permite al desarrollador acceder a los componentes de los dispositivos mediante programación a alto nivel.

Dado que Android es una plataforma que posee una pila de software, las funciones de cada capa utilizarán elementos pertenecientes a la capa inferior.

Las capas de las que consta dicha arquitectura son:

### Aplicaciones (Applications):

Esta capa está formada por aquellas aplicaciones con/sin interfaz de usuario, instaladas en la máquina Android (todas deben correr en Dalvik para garantizar la seguridad). Además, todas tienen mismo marco de aplicación para acceder a los servicios del SO, por lo que se pueden crear aplicaciones que usen los mismos recursos de las nativas, reemplazando cualquiera del teléfono.

Dichas aplicaciones pueden ser:

- Nativas: programadas en C/C++.

- Administradas: programadas en Java.

- Por defecto en el dispositivo móvil

- Instaladas por el usuario

- La aplicación Inicio (Home)

### Entorno de aplicación (Application Framework):

Esta capa la forman todas las clases y servicios usados para la realización de las funciones de las aplicaciones, apoyándose en bibliotecas Java de acceso mediante la máquina Dalvik, las librerías y el entorno de ejecución de Android. Ha sido diseñada para simplificar la reutilización de componentes (pudiendo reemplazarlos), siempre sujeta a restricciones de seguridad.

Algunas de las bibliotecas más importantes de esta capa son:

**- Administrador de actividades (Activity Manager):** Controla la pila de actividades, su ciclo de vida y proporciona un sistema de navegación entre aplicaciones.

**- Administrador de ventanas (Windows Manager):** Organiza lo que se muestra en la pantalla del dispositivo, creando “espacios” que pueden ser ocupados por las distintas actividades.

**- Proveedor de contenidos (Content Provider):** Mecanismo sencillo de encapsulación de datos para ser compartidos por distintas aplicaciones.

**- Sistema de vistas (View System):** Una vista es un elemento que ocupa un área en la pantalla y es la responsable del dibujo y la gestión de eventos; es la parte visual de los componentes. Es la clase base para los widgets utilizados para la construcción de interfaces de usuario, tales como botones, cuadros de texto, navegador web o visor de Google Maps.

**- Administrador de notificaciones (Notification Manager):** Permite a las aplicaciones notificar al usuario alertas en la barra de estado. También utiliza elementos como los LEDs o el vibrador del dispositivo.

**- Administrador de paquetes (Package Manager):** Permite la obtención de información de los paquetes instalados en el dispositivo, pudiendo gestionar la instalación de nuevos.

**- Administrador de telefonía (Telephony Manager):** Permite el acceso a la pila de telefonía del dispositivo.

**- Administrador de recursos (Resource Manager):** Permite el acceso y gestiona los recursos de una aplicación que están fuera del código, tales como imágenes, sonidos, disposiciones de las vistas dentro de una actividad (layouts) o cadenas de texto traducidas.

**- Administrador de ubicaciones (Location Manager):** Permite determinar la posición geográfica, gracias al GPS instalado, del dispositivo.

**- Servicio XMPP (XMPP Service):** Protocolo abierto y extensible basado en XML ideado para mensajería instantánea, estableciendo una plataforma para el intercambio de datos.

### Entorno de ejecución de Android (Android Runtime):

Formado por las librerías esenciales de Android (incluyen la funcionalidad de las librerías habituales de Java así como otras específicas de Android), está basado en el concepto de máquina virtual utilizado en Java. También se incluye el módulo “Core Libraries” con la mayoría de las librerías disponibles en el lenguaje Java.

De hecho, el componente principal es la máquina virtual Dalvik, que ejecuta todas y cada una de las aplicaciones no nativas de Android, codificándolas en Java, normalmente, y compilándolas en un formato especifico, permitiendo compilar una única vez las aplicaciones y distribuirlas (ya compiladas), sabiendo que se ejecutarán en cualquier dispositivo.

A diferencia de otras máquinas virtuales, Dalvik se basa en registros en lugar de una pila para almacenar los datos, requiriendo menos instrucciones y, por consiguiente, permitiendo ejecuciones más rápidas con menos recursos.

- Algunas características de Dalvik que facilitan la optimización de recursos son:

- Ejecución de ficheros .dex para ahorrar memoria, ya que la máquina virtual no es compatible con el bytecode de Java.

- Basada en registros.

- Cada aplicación correo en su propio proceso Linux y con su propia instancia en Dalvik.

- Delega al Kernel de Linux (núcleo) el manejo de la mamoria a bajo nivel y funciones como threading (hilos de ejecución).

Hay que destacar que a partir de Android 5.0 se sustituye la máquina virtual Dalvik por ART, consiguiendo reducir el tiempo de ejecución.

### Librerías (Libraries):

La siguiente capa que se sitúa justo sobre el Kernel de Linux la componen las bibliotecas nativas (librearías) de Android escritas en C/ C++ y compiladas para la arquitectura hardware específica del teléfono. Dichas librerías se encuentran instaladas en el terminal antes de ser puesto en venta y su objetivo es el de proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando de esta manera tener que codificarlas cada vez, y garantizando su eficiencia.

Entre las librerías nativas se encuentran:

- **Gestor de superficies (Surface Manager):** Encargado de componer los diferentes elementos de navegación de pantalla a partir de capas gráficas 2D y 3D. Gestiona también las ventanas de las distintas aplicaciones activas en cada momento.

- **Bibliotecas multimedia (Media Framework):** Librería basada en OpenCore; soporta códec de visualización, reproducción y grabación de multitud de formatos de imagen, vídeo y audio (JPG, GIF, PNG, MPEG4, AVC (H.264), MP3, AAC o AMR).

- **SQLite:** Potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.

- **OpenGL|ES:** Motor gráfico 3D basado en las APIs de OpenGL 1.0 y 2.0 de Android.

- **FreeType:** Permite mostrar fuentes tipográficas tanto basadas en mapas de bits (bitmap) como en renderizado vectorial.

- **LibWebCore:** Soporta un moderno navegador web utilizado en el navegador Android y en la vista webview.

- **SGL (Scalable Graphics Library):** Motor gráfico de 2D de Android.

- **SSL (Secure Sockets Layer):** Proporciona servicios de encriptación al acceder a Internet por medio de criptografía.

- **Biblioteca del sistema (Libc):** Proporciona funcionalidades básicas para la ejecución de las aplicaciones.

### Núcleo de Linux (Linux Kernel):

El núcleo del sistema operativo Android está basado en el kernel de Linux, versión 2.6 (o 3.0, según la versión). Actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura a los que acceden las aplicaciones. Para cada elemento hardware del dispositivo existe un controlador (*driver*) que permite utilizarlo desde el software (el desarrollador no acede directamente a esta capa, sino que utiliza las librerías disponibles en capas superiores). También proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso y la pila de protocolos, y se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación, etc.

Como es el primer estrato en la arquitectura, y la base de la estructura las características básicas del primero, dependen en gran medida del segundo nivel, puesto que si un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro del núcleo de Linux incluido en el propio Android.

## 2.4.5. VERSIONES

En las siguientes líneas se describen las versiones lanzadas hasta la fecha actual, identificándose por versión, número de API y el nombre con el que se le conoce (nombres de postres).

Antes de empezar un proyecto, elegir la versión del sistema para la que se desea realizar la aplicación es tan importante como escoger el SO del dispositivo, ya que la disponibilidad de las clases y los métodos de Android varía a partir de una versión.

Existen las versiones pre-comerciales (Alfa y Beta) de Android y las versiones oficiales. Dado el elevado número de versiones y la antigüedad con respecto a la última versión lanzada al mercado, haremos un breve análisis de las últimas y más usadas hasta la fecha.

### Android 4.4.X KitKat (API 19)


*Figura 3 - Versión Android 4.4.X. Kit Kat*

KitKat [2] no ofrecía una lista enorme de cambios radicales en la funcionalidad como Ice Cream Sandwich. En cambio, su propósito principal fue la de incorporar las últimas versiones a todos los dispositivos Android (ya fueran de gama baja, media o alta). Gracias a esto, Google redujo el SO para que pueda ejecutarse en muchos más dispositivos, lo que ayuda a cerrar la brecha entre los dispositivos.

Las características más importantes [3] de esta versión son:

1. Google Now es un asistente que escucha lo que preguntes y encontrará lo que estás buscando. Con esta aplicación hablarás a tu smartphone para hacer búsquedas, enviar mensajes, realizar llamadas a contactos o cualquier otra tarea común. Podrá ser activado bajo el comando de voz *Ok Google*, o tocando el icono del micrófono.

2. Con el identificador inteligente de llamadas si recibes una llamada de un número que no está en tus contactos, tendrás la posibilidad de ver no sólo el número, sino hará una búsqueda de números que coincidan con negocios que estén registrados en *Google Maps*.

3. KitKat añade una mejora a la pantalla completa con "inmersive mode" (cuando el usuario la activa, la interfaz de usuario se oculta automáticamente).

4. Por medio de Chromecast podrás ver tus películas y series favoritas en tu televisión HD.

### Android 5.X Lollipop (APIs 21 y 22)


*Figura 4 - Versión Android 5.X Lollipop*

En esta nueva versión [4] no encontraremos una inserción enorme de funcionalidades con respecto a KitKat. Sin embargo, lo que cambia enormemente es el aspecto visual gracias a que la base de diseño de esta versión es Material Design e integrando, por primera vez, Android Runtime (entorno de ejecución que reemplaza a Darvik).

Las características [5] más importantes de esta versión son:

1. Google incluyó de forma oficial la “linterna” a su sistema operativo.

2. Mientras el dispositivo está bloqueado, podemos ver e interactuar con las notificaciones de las aplicaciones.

3. Con esta versión, podemos conocer cuánto tiempo durará la batería o cuánto tiempo hay que esperar hasta que esté completamente cargada.

4. Incluye un modo para ahorrar batería que se puede configurar para que funcione al quedar 15% o 5% de batería, o nunca si se prefiere.

### Android 6.0.X Marshmallow (API 23)


*Figura 5 - Versión Android 6.0.X Marshmallow*

Visualmente, no ofrece grandes cambios como lo hizo la anterior versión; sin embargo, la interfaz tiene algunos cambios sutiles que buscan mejorar la experiencia de los usuarios [6].

Las aplicaciones están organizadas alfabéticamente. Además, nos ofrece la posibilidad de interactuar más fácilmente con los permisos que le damos a las aplicaciones.

Las características [7] más importantes de esta versión son:

1. Las aplicaciones piden consentimiento del usuario cada vez que se quiera usar una función.

2. Esta versión incluye Doze para la gestión de la batería. Éste, en determinado momento, cuando no uses el teléfono y este se encuentre bloqueado, el sistema activará este modo de ahorro para que puedas disfrutar de tu batería por más tiempo. Se reducen todos los procesos, pero no se los mensajes importantes y las llamadas entrantes.

3. Tenemos la opción de comprobar el consumo de memoria, ver qué aplicaciones están acaparando más recursos y comprobar las estadísticas de las últimas 3, 6, 12 o 24 horas para mejorar nuestros hábitos y el rendimiento del equipo.

4. Ahora, después de un reinicio de fábrica, podremos hacer regresar las aplicaciones como por arte de magia, así como todas aquellas preferencias y personalizaciones que hubieramos marcado.

### Android 7.X Nougat (API 24 y 25)


*Figura 6 - Versión Android 7.X Nougat*

En esta nueva versión, Google trae importantes cambios a su sistema operativo contando con más de 250 novedades.

La diferencia principal con Mashmallow [8] es que, mientras ésta sólo fue un refinamiento de los cambios que se lograron con Lollipop, Nougat es un cambio importante en el diseño y funciones del sistema operativo.

Las características [9] más importantes de esta versión son:

1. Uso de doble ventana, todo desde la misma pantalla y sin tener que estar cambiando de ventana para ver una u otra aplicación.

2. Personalización de la barra de navegación mediante System UI Turner, con la que puedes modificar ciertos aspectos de la interfaz (añadir botones o modificar la barra de navegación).

3. En esta versión se incluye un botón de multitarea que permite cerrar todas las aplicaciones abiertas en segundo plano.

4. Mayor control de datos, en especial para usuarios con tarifas reducidas, gracias a un economizador de datos que restringe el uso de ciertas aplicaciones.

### Android 8.0 Oreo (API 26)

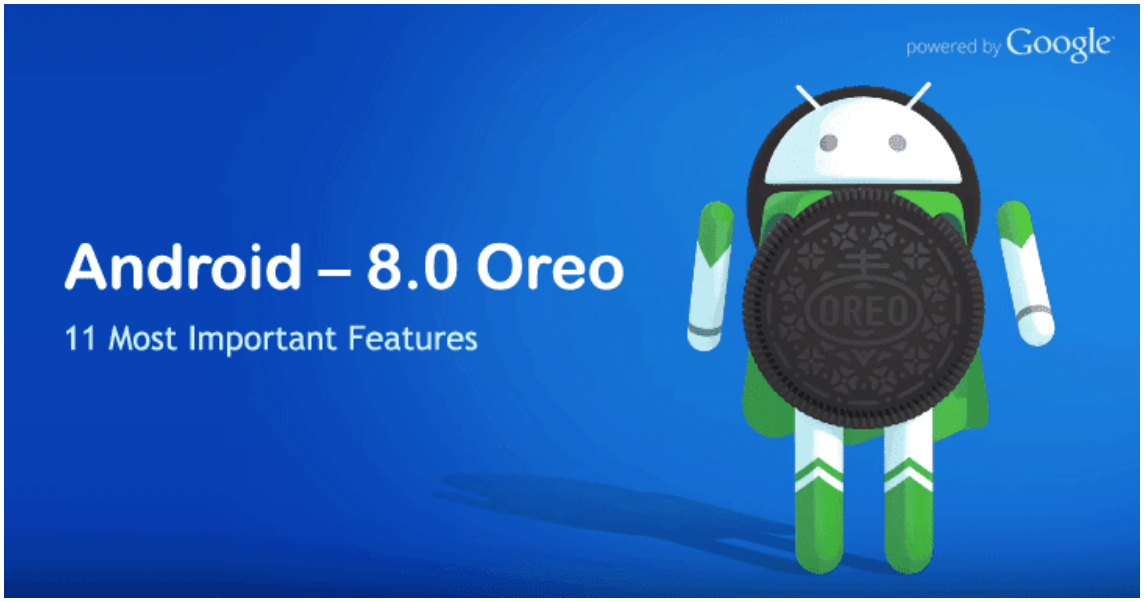



*Figura 7 - Versión Android 8.0 Oreo*

Esta es la última y más actual versión creada por Google [10] y está orientada a crear un "puente" entre el sistema operativo y los fabricantes.

Con el nuevo proyecto utilizado llamado Project Treble, se quiere separar el vendor del sistema Android, para reducir el tiempo que les toma a los fabricantes actualizar sus dispositivos [11]. Además, destacar que cada vez hay menos motivos para cambiar la ROM a un dispositivo ya que los fabricantes han optimizado el desarrollo sus capas de personalización.

Las características [12] más importantes de esta versión son:

1. Si recibimos una notificación podemos posponerla para recibirla de nuevo 15 minutos, 30 minutos, 1 hora o 2 horas más tarde.

2. Habrá jerarquía en la barra de notificaciones, mostrándose primero las notificaciones en curso, luego las enviadas entre personas, a continuación las notificaciones generales y, finalmente, las menos importantes como tiempo, promociones.

3. Con esta versión, Si desactivamos el Wi-Fi fuera de casa para ahorrar batería podemos decirle al sistema que active automáticamente el Wi-Fi cuando lleguemos a casa o detecte una red guardada.

4. Compatibilidad con múltiples pantallas para poder conectar nuestro dispositivo a una pantalla externa para enviar lo que estamos viendo u otra actividad. En la pantalla del móvil veremos una actividad en la resolución de nuestro dispositivo y la otra actividad a la resolución de la pantalla.

### 2.4.6. ¿QUÉ VERSIÓN ESCOGER?

Cada vez que se comienza una nueva aplicación en Android hay un dilema común, ¿qué versión es mejor para programar?

Ante tal abanico de versiones, es difícil contestar a esa pregunta. Sin embargo, deberemos hacer una distinción clara: Qué aplicación quiero hacer y para qué tipo de usuario.

Si bien es cierto que si queremos realizar un proyecto para usarse con tecnologías modernas, deberemos usar la versión más cercana a la actual [13]. Por el contrario si queremos programar una aplicación que esté destinada a todo tipo de usuario y que no importe la versión utilizada, nos iremos a versiones más antiguas.

Dado que esta aplicación está orientada a niños pequeños (e indirectamente a sus padres jóvenes), la versión a utilizar será la destinada a dispositivos que no son ni muy antiguos ni muy nuevos (o con versiones nuevas). Por tanto, la elección de la versión para este proyecto será la versión 6.0 (API 23) o, por el nombre que mejor se la conoce, Marshmallow y que engloba casi el 40% de los dispositivos Android actuales.

## 2.5. HERRAMIENTAS PARA PROGRAMAR EN ANDROID

El número de herramientas existentes para programar en Android es muy elevado; sin embargo, a continuación se expondrán las más utilizadas actualmente y cuál de ellas es la elegida para realizar el proyecto.

### 2.5.1. ANDROID STUDIO

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android. Además del potente editor de códigos y las herramientas para desarrolladores, ofrece aún más funciones que aumentan la productividad durante la compilación de apps para Android [14].

Dado que su utilidad, en la actualidad, es muy elevada y en constante aumento, cada día se reportan y solucionan múltiples fallos, lo que provoca que su actualización sea continua.

Sus características [15] más importantes son:

- Se trata de un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.

- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.

- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.

- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.

- Gran cantidad de herramientas y frameworks de prueba.

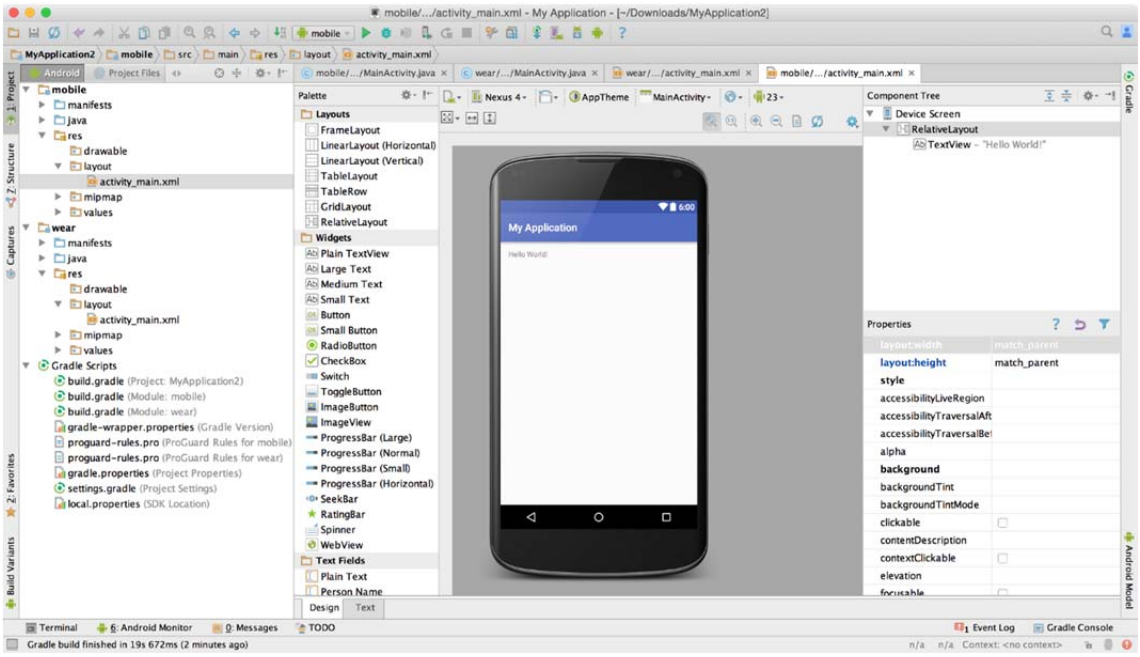



*Figura 8 - Entorno de trabajo de Android Studio*

## 2.5.2. APP INVENTOR

App Inventor es una aplicación que permite crear aplicaciones para sistemas operativos Android. Google lanzó su primera versión en 2010 bajo el nombre Google App Inventor, con el objetivo de poder ofrecer una solución que permitiera crear nuevas aplicaciones de software a usuarios sin conocimientos previos de programación. Tras unos cuantos años y unas cuantas actualizaciones, MIT App Inventor (su nombre actual) conjuga sencillez y potencia en una herramienta de desarrollo de software para uso particular y empresarial [16].

Sus características [17] más importantes son:

- El editor de bloques utiliza la librería Open Blocks de Java para crear un lenguaje visual a partir de bloques. Estas librerías están distribuidas bajo licencia libre.

- Permite crear una aplicación en una hora o menos, y se pueden programar aplicaciones más complejas en mucho menos tiempo que con los lenguajes más tradicionales.

- Se trata de una herramienta de código abierto que pretende realizar la programación y la creación de aplicaciones accesibles a una amplia gama de usuarios.

- Su interfaz gráfica le permite al usuario crear aplicaciones con muchas funcionalidades al alcance de unos cuantos clicks (se abre una gran puerta para muchas personas sin conocimientos de programación)

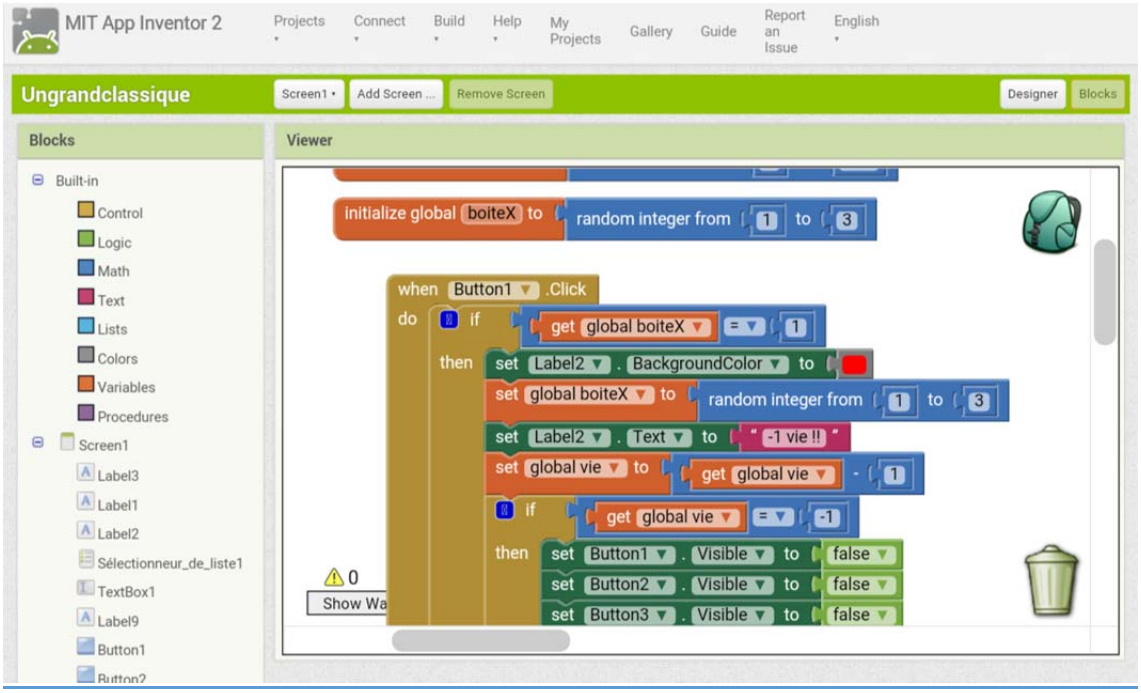



*Figura 9 - Entorno de trabajo de App Inventor*

## 2.5.3. ECLIPSE

Se trata de un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones. Esta plataforma, fundamentalmente, ha sido utilizada para desarrollar entornos de desarrollo integrados (IDE) como el de Java.

Debido a su arquitectura, es camaleónicamente adaptable y puede adaptarse a distintos entornos. En nuestro caso, para poder adaptarlo a Android, sería necesario cargarle un plugin específico llamado Android Development Tools (ADT) y estaría listo para poder usar Eclipse como aplicación para programar proyectos en Android [18].

Sus características [19] más importantes son:

- El entorno de desarrollo integrado de Eclipse emplea módulos para proporcionar toda su funcionalidad.

- Es uno de los IDE's más robustos dándonos la libertad de configurar nuestros propios ambientes instalando lo que realmente necesitamos.

- Eclipse funciona como un ambiente integrado sin ser necesidad de instalación (facilidad en el uso de plugins).

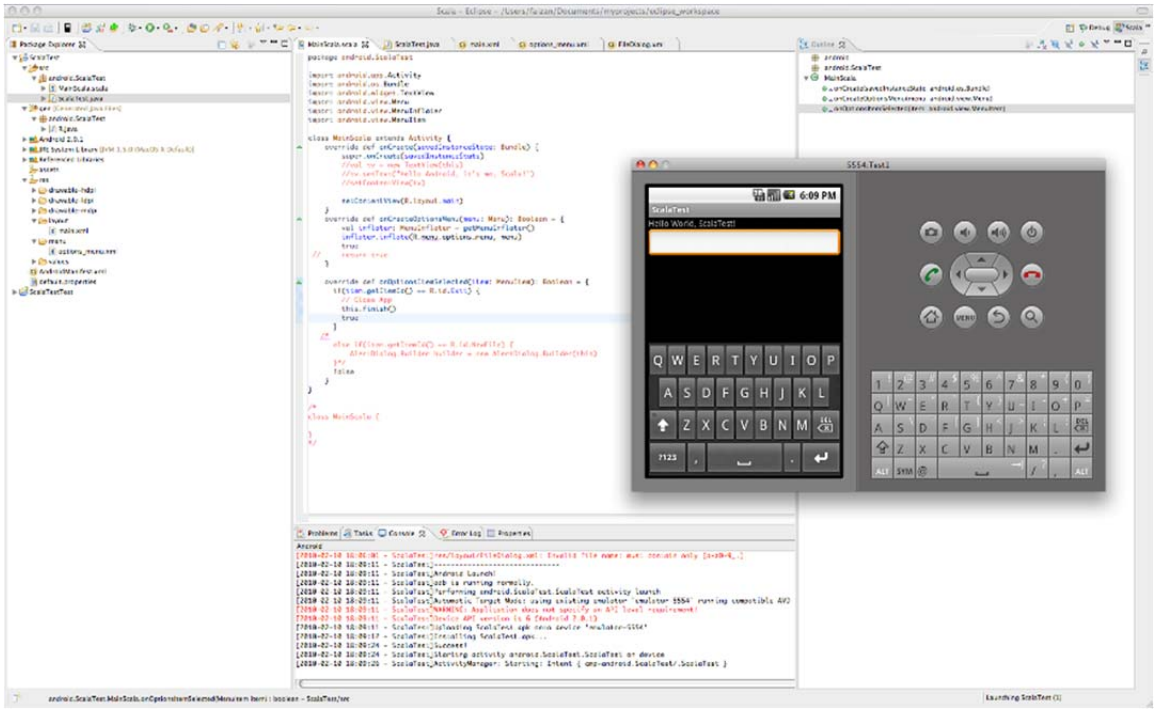


*Figura 10 - Entorno de trabajo de Eclipse*

## 2.5.4. UNITY 3D

Unity 3D es una de las plataformas para desarrollar videojuegos más completas que existen. Permite la creación de juegos para múltiples plataformas a partir de un único desarrollo, incluyendo el desarrollo de juegos para videoconsolas, ordenadores, móviles y tabletas [20].

Es posiblemente la tecnología de mayor crecimiento en estos momentos. Su principal limitación es el precio de su licencia completa.

Sus características [21] más importantes son:

- El editor permite agrupar rápidamente todas las escenas en un espacio de trabajo, mediante el uso de un editor intuitivo y fiable.

- Desarrollo de videojuegos de gran calidad, en pocos pasos, que se adaptan a todo tipo de resoluciones.

- Posibilita la publicación en numerosas plataformas, sin realizar ninguna tarea de implementación extra.

- Control exhaustivo de los recursos consumidos, con una ventana

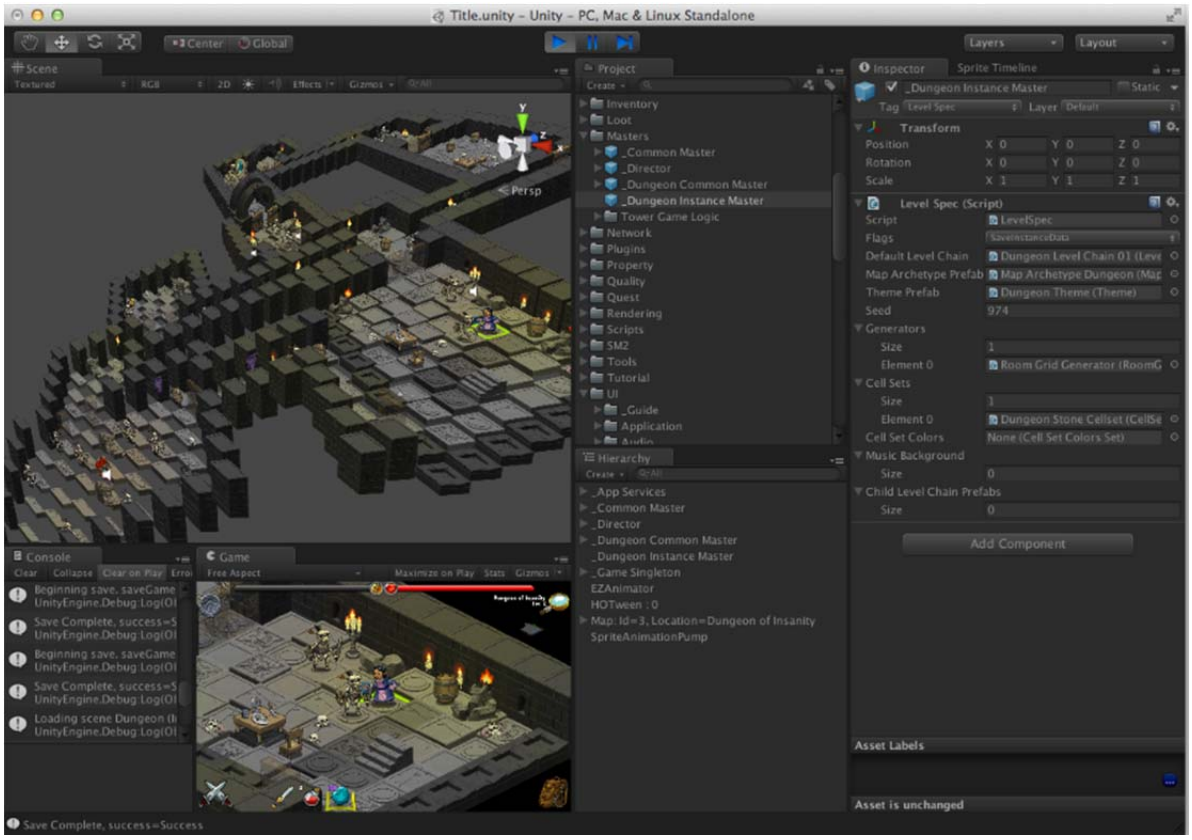



*Figura 11 - Entorno de trabajo Unity 3D*

## 2.6. ¿POR QUÉ ANDROID STUDIO?

Dado que Eclipse y Android Studio son los entornos más utilizados para programar aplicaciones en Android, expondré en las siguientes líneas las razones por las que he decidido que la herramienta usada para realizar la aplicación sea Android Studio [22].

*Figura 12 - Android Studio VS Eclipse*

¿Con qué entorno programo mi aplicación, Android Studio o Eclipse? Esta es la gran pregunta a la hora de comenzar el trabajo; sin embargo, el primero tiene unas cuantas razones de peso por el que ser elegido frente al segundo:

- La primera razón es la más sencilla y a la vez la más importante: Android Studio es puramente Android, creado para programar en Android.

- Se actualiza constantemente.

- Eclipse está a punto de desaparecer ya que, debido a esta actualización continua, se ha dejado de dar soporte al entorno.

- Posee internamente el compilador Gradle, lo que se traduce en mayor rapidez, mejor rendimiento y exporta los .apk con mayor facilidad

- No hay dependencias y se puede crear cualquier tipo de emulador.

¿Por qué programar con otra herramienta, si existe una creada específicamente para ello? Por todas estas razones, el entorno elegido para programar la aplicación será Android Studio.


*Figura 13 - Android Studio como herramienta de programación*

## 2.7. SISTEMAS DE DIÁLOGO

Los sistemas de diálogo reciben como entrada frases de forma oral y general, como salida, frases de forma, también, oral. Su función es la de emular el comportamiento de una persona al realizar una acción.

Mediante iteraciones entre la persona y el programa, estos sistemas facilitan la interacción entre usuario y máquina sin necesidad de teclado o el aspecto táctil, tal y como indica la figura 14.

Tras cada intervención del usuario, el sistema realiza una serie de acciones como respuesta, repitiéndose cíclicamente

- Reconoce las secuencias de palabras (habla) del usuario

- Extrae el significado de cada palabra (interpretación semántica)

- Realiza operaciones o almacena la información en bases de datos

- Realiza la acción adecuada tras la operación y genera la cadena de texto

- Reproduce el mensaje hablado tras la conversión de la cadena

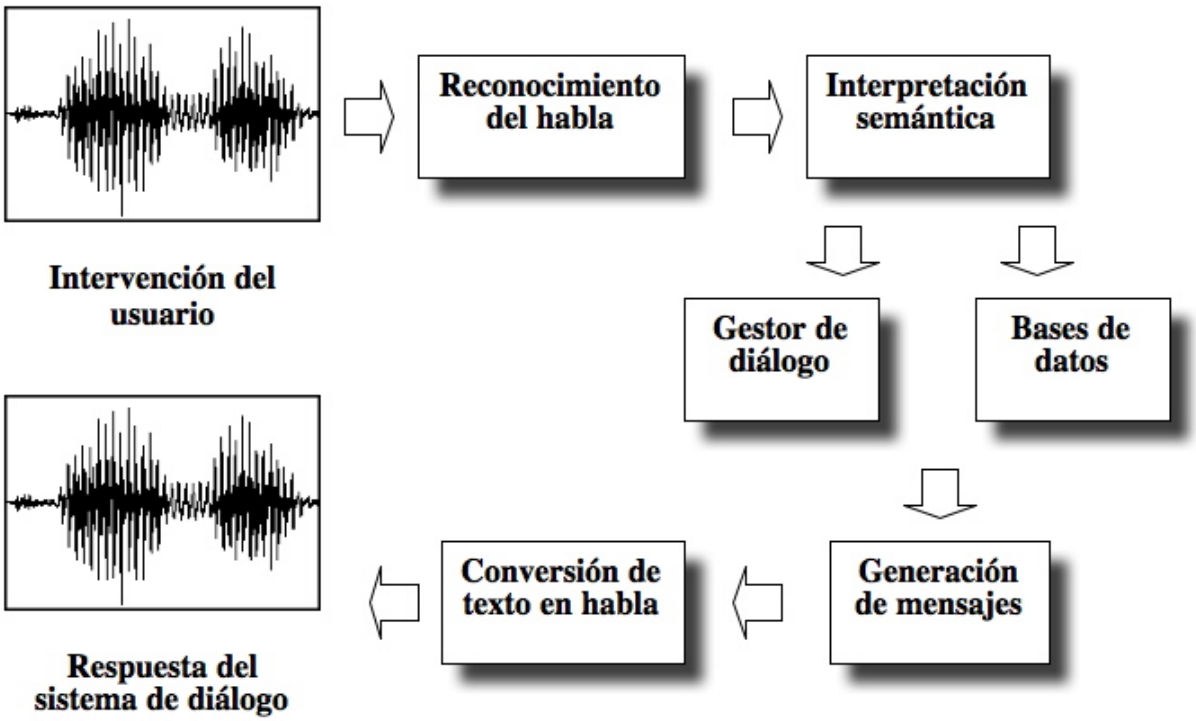



*Figura 14 - Esquema de un sistema de diálogo*

Estos estos sistemas tienen algunos puntos débiles difíciles de solventar:

- El vocabulario disponible de algunos sistemas de diálogo es escaso.

- Casi siempre existirá ruido de fondo, interferencias o cualquier sonido no deseado.

- Dado el variado número de dialectos para un mismo idioma, es difícil conseguir la perfección de entendimiento de estos sistemas.

A pesar de estos inconvenientes, los sistemas de diálogo son una gran herramienta capaz de resolver muchos problemas humanos y cada día se mejoran considerablemente.

## 2.8. TEXT TO SPEECH en Android (TTS en Android)

### 2.8.1. INTRODUCCIÓN

Desde la versión 1.6, Android incorpora una nueva característica llamado TTS (o síntesis de voz) que permite al dispositivo "hablar" en diferentes idiomas. Mediante el motor TTS, las aplicaciones son capaces de producir voz a partir de cualquier cadena de texto, permitiendo al usuario interactuar con la aplicación sin la necesidad de leer la pantalla del dispositivo [23].

Para poner el motor de síntesis de voz de Google como predeterminado tenemos que ir a Ajustes > Idioma y entrada de texto > Opciones de texto a voz y seleccionar "Síntesis de Google", como demuestra la figura 15:

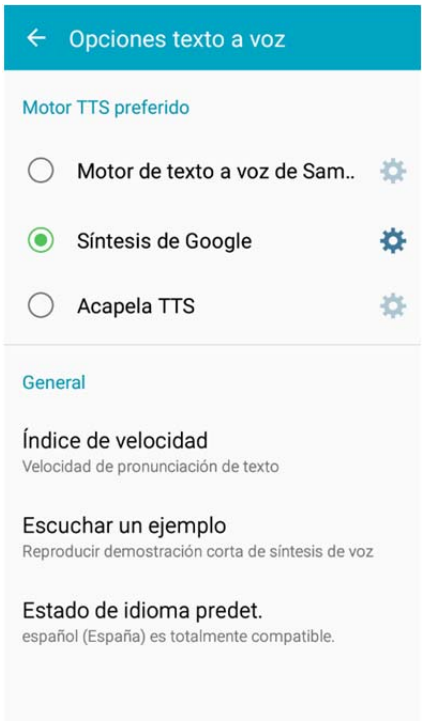



*Figura 15 - Opciones de texto a voz*

Para poder escuchar un ejemplo, una vez seleccionado el motor preferido, seleccionaremos "Escuchar un ejemplo". La actividad correspondiente inicializa el motor de síntesis configurándolo a la velocidad e idioma establecidos, enviando un texto al motor, y éste se encarga de reproducirlo en la salida de audio.

Las opciones disponibles, además de la ya mencionada, son:

**- Escuchar un ejemplo:** Seleccionando esta opción, se escucha un ejemplo del motor de síntesis en el idioma que se encuentre seleccionado en la opción Idioma.

**- Índice de velocidad:** Permite seleccionar la velocidad a la que se lee el texto entre los valores Muy lento y Más rápido.

**- Estado de idioma predet:** Señala si el motor de síntesis es compatible con el idioma seleccionado.

Además de aportar simplicidad, el motor TTS apenas consume memoria y espacio en disco y su coste es cero. El inconveniente es que, en comparación con otros motores "de pago" disponibles en Android, la calidad de la voz es regular, ya que resulta algo monótona y robótica.

Independientemente del número de motores de síntesis de voz que se encuentren instalados en el dispositivo, únicamente habrá un motor de síntesis de voz activo que será compartido por todas las actividades que quieran hacer uso de él. Por ello, una actividad no sabrá el instante en el que se escuchará el texto que se envié al motor (Hashimi y cols).

## 2.8.2. INCORPORACIÓN DE TTS EN UNA APP ANDROID

La lógica del motor de síntesis de voz es complicada, sin embargo, ahora es posible desarrollar apps que se encarguen de enviar el texto al motor para que lo reproduzca, sin conocer su lógica interna (Hashimi y cols).

Para ello, se debe utilizar el paquete android.speech.tts de la API de Android, en particular la clase android.speech.tts.TextToSpeech que permite introducir en una cola el texto que se quiere reproducir. La opción más simple es la de enviar el texto al motor de síntesis mediante la utilización de dicha clase.

A continuación, se resume las clases e interfaces que componen el paquete android.speech.tts y android.speech.tts.TextToSpeech en la versión más actualizada y los aspectos a tener en cuenta a la hora de integrar el TTS en una app Android.

### 2.8.2.1. Paquete android.speech.tts

**INTERFACES:**

| Interfaz | Definición |
|---|---|
| **SynthesisCallback** | Método que recibe el resultado de la síntesis de texto a voz realizada por el motor de síntesis. |
| **TextToSpeech.OnInitListener** | Método invocado cuando finaliza la inicialización del motor de síntesis. |

*Tabla 1 - Interfaces android.speech.tts*

### CLASES:

| Clase | Definición |
|---|---|
| **SynthesisRequest** | Contiene los datos que requiere el motor para realizar la síntesis de texto a voz. |
| **TextToSpeech** | Se encarga de la síntesis de texto a voz. El resultado de la síntesis será reproducido o guardado en un fichero de audio. |
| **TextToSpeech.Engine** | Contiene las constantes y nombres de parámetros que se utilizan para el control de la síntesis de texto a voz. |
| **TextToSpeech.EngineInfo** | Solicita la información del motor de síntesis de texto a voz instalado en el dispositivo. |
| **TextToSpeechService** | Clase abstracta que permite implementar los métodos del motor de síntesis, tales como: onIsLanguageAvailable, onLoadLanguage, onGetLanguage, onSynthesizeText, onStop. |
| **UtteranceProgressListener** | Llamada invocada en eventos relacionados con el progreso de un enunciado a través de la cola de síntesis (cada enunciada está asociado a un "speak" y a un identificador según KEY_PARAM_UTTERANCE_ID). |
| **Voice** | Controla los eventos relacionados con el progreso de una cadena de texto dentro de la cola que contiene todas las cadenas a ser sintetizadas. |

*Tabla 2 - Clases android.speech.tts*

#### 2.8.2.2. Clase android.speech.tts.TextToSpeech

**CLASES ANIDADAS:**

| Clase anidada | Definición |
|---|---|
| **TextToSpeech.Engine** | Clase que proporciona los nombres y las constantes para controlar el TTS. |
| **TextToSpeech.EngineInfo** | Clase que proporciona información sobre el motor TTS instalado. |
| **TextToSpeech.OnInitListener** | Interfaz que invoca a una llamada tras completar la inicialización del motor TTS. |

*Tabla 3 - Clases anidadas android.speech.tts.TextToSpeech*

**CONSTANTES:**

| Constante | Definición |
|---|---|
| **ACTION_TTS_QUEUE_PROCESSING_COMPLETED** | Acción de broadcast en la que el sintetizador TTS ha completado el procesamiento de todo el texto de la cola de voz. |
| **ERROR** | Muestra un fallo genérico de operación. |
| **ERROR_INVALID_REQUEST** | Muestra un fallo causado por una petición incorrecta. |
| **ERROR_OUTPUT** | Muestra un fallo relacionado con un dispositivo o un archive de audio. |
| **ERROR_SERVICE** | Muestra un fallo con el servicio TTS. |
| **LANG_AVAILABLE** | Permite conocer si el uso de un lenguaje en cuestión es permisible. |
| **LANG_NOT_SUPPORTED** | Muestra que el lenguaje en cuestión no es admisible. |
| **QUEUE_ADD** | Modo en el que la nueva entrada se coloca al final de la cola. |
| **QUEUE_FLUSH** | Modo en el que, al introducir una entrada, se borran todos los datos pertenecientes a la cola. |
| **STOPPED** | Muestra una parada tras petición del usuario. |
| **SUCCESS** | Muestra una operación exitosa. |

*Tabla 4 - Constantes android.speech.tts.TextToSpeech*

### CONSTRUCTORES PÚBLICOS

| Constructor |
|---|
| **TextToSpeech(Context context, TextToSpeech.OnInitListener listener)** |
| **TextToSpeech(Context context, TextToSpeech.OnInitListener listener, String engine)** |

*Tabla 5 - Constructores públicos android.speech.tts.TextToSpeech*

### MÉTODOS PÚBLICOS

| Método | Definición |
|---|---|
| **addSpeech** | Añade un "mapping" entre una cadena/CharSequence y un archive de audio. |
| **getAvailableLenguages** | Petición sobre los lenguajes disponibles el motor TTS. |
| **getDefaultEngine** | Devuelve el nombre del paquete de voz genérico del motor TTS. |
| **getDefaultVoice** | Devuelve la referencia de la voz genérica del lenguaje genérico del motor TTS. |
| **getEngines** | Devuelve una lista de todos los motores TTS instalados. |
| **getVoice** | Devuelve la referencia de la voz usada actualmente por el motor TTS. |
| **isSpeaking** | Comprueba si el motor TTs está reproduciendo sonido. |
| **playSilentUtterance** | Reproduce un "silencio" durante el tiempo especificado, usando un modo de cola descrito. |
| **setLanguage** | Determina el lenguaje del motor TTS. |
| **setPitch** | Determina el tono de la voz; podrá ser más agudo o más grave según se especifique. |
| **setVoice** | Determina la voz del motor TTS. |
| **speak** | Reproduce un texto usando un modo de cola y parámetros especificados. |
| **Stop** | Interrumpe el sonido reproducido y elimina cualquier otro elemento en la cola. |

*Tabla 6 - Métodos públicos android.speech.tts.TextToSpeech*

### 2.8.3. ALTERNATIVAS A GOOGLE TTS

Tras realizar una búsqueda en Google Play de motores de síntesis de voz, hemos encontrado varias aplicaciones tanto de pago como gratuitas. En la tabla 7 se enumeran y detallan, en cada una de ellas, la calidad que ofrecen al usuario, además de si son o no gratuitas.

| APLICACIONES | CALIDAD | PRECIO | CONTIENE ESPAÑOL |
|---|---|---|---|
| **Google TTS** | Se trata de la aplicación de síntesis de voz general para cualquier dispositivo Android. La calidad es bastante buena, un poco monótona, pero se asemeja mucho a una voz real. | GRATUITO | SI |
| **Samsung TTS** | Se trata del motor de búsqueda propio de los dispositivos Samsung y sólo dedicado a ellos. La calidad que ofrece es bastante aceptable, un poco más robótica que la de Google TTS. | GRATUITO | SI |
| **CMU Flite Text** | El peor motor de síntesis de los expuestos en la tabla. Además, la voz es muy robótica y no tiene paquetes de español. | GRATUITO | NO |
| **Voz TTS Vocalizer** | Ofrece una calidad regular. Su principal inconveniente es que ofrece una calidad deplorable para ser una aplicación de pago | DE PAGO | SI |
| **vnSpeak TTS** | Motor gratuito de síntesis de voz muy robótico. Su principal inconveniente es que no existe ningún paquete de voz en español. | GRATUITO | NO |

| | | | |
|---|---|---|---|
| **Acapella TTS Voices** | Ofrece la mejor calidad de todos los motores contenidos en esta tabla. Su principal inconveniente es que hay que pagar por cada voz que se quiere descargar. | DE PAGO | SI |
| **Voxygen Expensive Speech** | Ofrece una calidad bastante aceptable, con sólo dos voces (una de hombre y otra de mujer). Su inconveniente principal es, a parte que es de pago, el mencionado antes del escaso número de voces distintas. | DE PAGO | SI |
| **Cereproc** | Posee una calidad muy buena, comparándose con los antiguos IVONA y SVOX | DE PAGO | NO |

*Tabla 7 - Motores de síntesis de voz en la actualdiad*

Tras esta explicación habrá que señalar que el motor de síntesis a utilizar será "Google TTS" por las siguientes razones:

- Dado que hemos elegido concretamente Android como Sistema Operativo debido a que es de coste cero, no tendría sentido que el motor TTS fuese de pago.
- Es el motor común para todos los dispositivos Android; da igual la marca y el modelo.
- La calidad es bastante buena y está en constante actualización y mejora.

# 2.9. RECONOCIMIENTO DE VOZ

## 2.9.1. INTRODUCCIÓN

El reconocimiento automático de voz (RAH) tiene como objetivo permitir la comunicación hablada entre seres humanos y dispositivos.

Un sistema de reconocimiento de voz es capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes sobre un proceso, como indica la figura 16.

Estos sistemas se llevan usando desde hace mucho tiempo en ordenadores, principalmente para poder escribir en un documento mediante el habla por un micrófono externo. Actualmente, muchos de los programas de edición de texto ya incorporan en sus últimas versiones el sistema de reconocimiento de voz integrado.

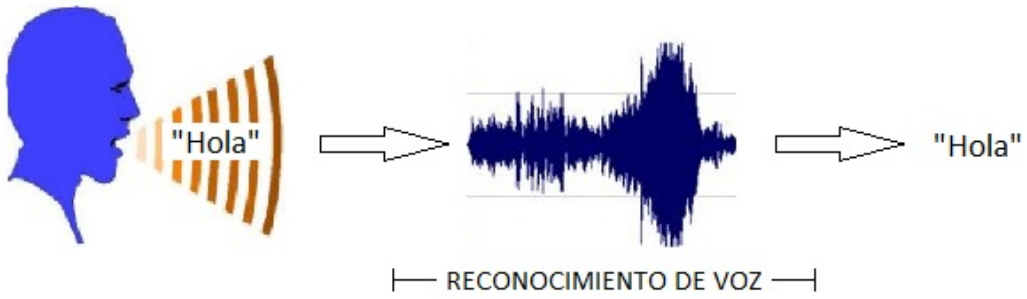



*Figura 16 - Esquema general del reconocimiento de voz*

Dado que Android es una plataforma abierta, las aplicaciones pueden utilizar cualquier servicio de reconocimiento de voz registrado en el dispositivo. Por ejemplo, la aplicación por defecto “Búsqueda por Google” responde a un RecognizerIntent mostrando una ventana con el título “Hable Ahora”, al igual que el botón micrófono del teclado Android.

Además, otra posibilidad para integrar el reconocimiento de voz, es mediante el SDK (Software Development Kit), disponiendo de 27 idiomas para dicho uso.

## 2.9.2. EJEMPLOS DE RECONOCEDORES DE VOZ

Hay muchos sistemas de reconocimiento actualmente en el mercado para todos los Sistemas Operativos. Sin embargo, expondremos a continuación una breve descripción sobre los 3 más importantes hasta la fecha:

### 2.9.2.1. Google Now

Google Now vio la luz en 2012 con Android Jelly Bean (4.1). Integra en una sola aplicación todos los servicios de Google, usando una interfaz de lenguaje natural, volcándose en la red para obtener la respuesta a cualquier petición del usuario [24].

Si, por ejemplo, queremos buscar un gimnasio cerca de nuestra casa, podremos hacerlo a mano en el dispositivo o mediante voz. La aplicación nos mostrará los gimnasios más cercanos desde nuestra ubicación; una vez elegido, nos podrá dirigir al gimnasio mediante Google Maps.

Además, se sincroniza con nuestra cuenta de Gmail y también nos lanza recordatorios de los eventos que hemos guardado en el calendario de Google, información sobre vuelos, el estado de nuestros pedidos en Amazon, cumpleaños de nuestros amigos de Google +, resultados deportivos, etc.

Es una herramienta muy potente que viene instalada por defecto en todos los terminales Android. Basta con decir "OK, Google", el teléfono se activa y a partir de ahí podemos hacerle cualquier pregunta o darle cualquier orden [25].

### 2.9.2.2. SIRI

Siri fue lanzada en el año 2007, y relanzada en el 2011 para el famoso iPhone 4S. Se trata de un asistente personal unificado que se encuentra disponible en iOS, tiene licencia exclusiva de Apple y cuenta con soporte para idioma Español [26].

Con Siri no tienes que aprenderte comandos especiales para hablar, ya que ella entiende el lenguaje coloquial.

Otra característica interesante, es que se adapta perfectamente con el iPad y puede desplegar mapas y guías de cualquier ciudad del mundo. Además, cuenta con acceso Bluetooth, lo que facilita la interacción con otros dispositivos.

Para usarlo, no se puede iniciar mediante voz. Es necesario dejar apretado el botón del dispositivo Apple.

#### 2.9.2.3. CORTANA

Cortana nació en 2014 con Windows Phone 8.1. Es un asistente personal inteligente desarrollado por Microsoft y utilizado en dispositivos con Windows como Sistema Operativo. Está disponible en modo programa para Android e iOS [27].

Su utilidad más interesante es su capacidad de realizar tareas sin que se lo pidas, aprendiendo de los hábitos del usuario, sus gustos y sus necesidades, y programa tareas automáticas sin que tengas que pensar en ello.

Se diferencia de Google Now y de Siri en que es el primer asistente virtual que incluye una "Libreta", donde se guarda toda la información que tiene sobre el usuario. De esa manera la información es fácilmente controlable, pudiendo borrar todo rastro o introducir datos para mejorar el servicio. También es el primer asistente que permite a los desarrolladores integrarse con él, permitiendo utilizar sus aplicaciones mediante la voz.

Para activarlo, sólo es necesario decir “Hola Cortana” y ya estará listo para ser usado.

### 2.9.3. INCORPORACIÓN DEL RECONOCIMIENTO DE VOZ EN UNA APP

Existen más posibilidades de integración, pero la más sencilla de todas consiste en enviar un objeto android.speech.RecognizerIntent a la aplicación “Búsqueda por Google.

A través del paquete android.speech, Android permite tanto integrar en una aplicación los servicios de reconocimiento de voz instalados por defecto, como diseñar los servicios de reconocimiento totalmente nuevos.

Aunque las aplicaciones no necesiten permisos para acceder a los servicios de reconocimiento de voz, es necesario disponer de conexión a Internet para que pueda funcionar.

Como hemos comentado, la opción más sencilla de integrar el reconocimiento de voz es lanzar un Intent (objeto) android.speech.RecognizerIntent, lo que activa el grabador de voz de Android solicitando la entrada por voz al usuario.

Dado que el fichero resultante se envía a procesar a los servidores Google, es necesario tener activada la conexión a Internet.

A continuación, se resume las interfaces y clases que componen el paquete android.speech.tts y las constantes y métodos de la clase android.speech.RecognizerIntent en la versión más actualizada a la hora de integrar el TTS en una app Android.

#### 2.9.3.1. Paquete android.speech

**INTERFACES:**

| Interfaz | Descripción |
|---|---|
| **RecognizerListener** | Se utiliza para recibir notificaciones desde el SpeechRecognizer cuando ocurren los eventos relacionados con el reconocimiento de voz. |

*Tabla 8 - Interfaces android.speech*

**CLASES:**

| Clase | Definición |
|---|---|
| **RecognitionService** | Proporciona una clase base para implementaciones de servicios de reconocimiento. |
| **RecognitionService.Callback** | Recibe llamadas desde el servicio de reconocimiento de voz y las devuelve al usuario. |
| **RecognizerIntent** | Constantes para ayudar al reconocimiento de voz mediante el inicio de un Intent. |
| **RecognizerResultsIntent** | Constantes para los Intents relacionados con la presentación de los resultados del reconocimiento de voz. |
| **SpeechRecognizer** | Proporciona acceso al servicio de reconocimiento de voz. |

*Tabla 9 - Clases android.speech*

### 2.9.3.2. Clase android.speech.RecognizerIntent

**CONSTANTES:**

| Constante | Definición |
|---|---|
| **ACTION_GET_LANGUAGE_DETAILS** | Invoca un Intent de tipo broadcast que se envía a un objeto BroadcastReceiver con el fin de solicitar la lista de los idiomas disponibles del reconocimiento de voz. |
| **ACTION_RECOGNIZE_SPEECH** | Inicia la actividad encargada de solicitar al usuario que realice su consulta vía voz y de enviarla al Recognizer. |
| **ACTION_VOICE_SEARCH_HANDS_FREE** | Inicia la actividad encargada de solicitar al usuario que realice su consulta vía voz y de enviarla al Recognizer, sin requerir una acción de entrada del usuario. |
| **EXTRA_LANGUAGE** | Indica al Recognizer que realice el reconocimiento de voz en el idioma especificado por un código de idioma IETF (en español sería “es-ES”). |
| **EXTRA_LANGUAGE_MODEL** | Informa al reconocedor sobre el modelo del lenguaje utilizado por el ACTION_RECOGNIZE_SPEECH. |
| **EXTRA_RESULTS** | Muestra los resultados del reconocimiento de voz de tipo ArrayList<String> cuando se ha realizado el Intent del ACTION_RECOGNIZE_SPEECH. |
| **EXTRA_SUPPORTED_LANGUAGES** | Es un ArrayList<String> y contiene los idiomas disponibles en la implementación actual del reconocimiento de voz. |
| **RESULT_AUDIO_ERROR** | Devuelve un código tras detectarse error en el audio. |
| **RESULT_NO_MATCH** | Devuelve un código al no encontrar ninguna coincidencia con la palabra que dice el usuario. |

*Tabla 10 - Constantes clase android.speech.RecognizerIntent*

### MÉTODOS PÚBLICOS:

| Método | Definición |
|---|---|
| **getVoiceDetailsIntent(Context context)** | Devuelve un Intent de broadcast lanzado por sendOrderedBroadcast (permite el acceso a recursos y clases específicos de una aplicación, así como llamadas ascendentes) y usado para recibir detalles del paquete que implementa la búsqueda por voz. |

*Tabla 11 - Métodos públicos clase android.speech.RecognizerIntent*

# 2.10. BASES DE DATOS

## 2.10.1. INTRODUCCIÓN

Una base de datos (o BBDD) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Dado que distintos programas y usuarios deben utilizar los mismos datos, el concepto de base de datos generalmente está relacionado con el de red. Se habla de un "Sistema de información" para designar a la estructura global que incluye todos los mecanismos para compartir datos.

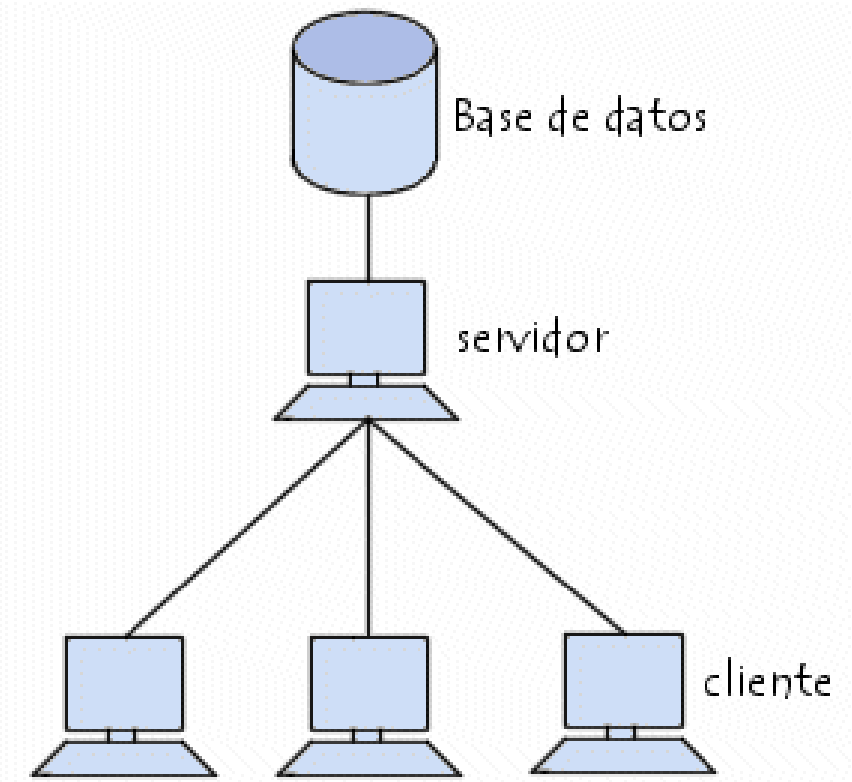



*Figura 17 - Esquema base de datos general*

Una base de datos proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, en concordancia con los derechos de acceso que se les hayan otorgado. Puede ser local (interna), es decir que puede utilizarla solo un usuario en un equipo, o puede ser distribuida (externa), es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red. La principal ventaja de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo [28].

Entre sus numerosas características [29], destacaremos.

- Independencia lógica y física de los datos.

- Redundancia mínima.

- Acceso concurrente por parte de múltiples usuarios.

- Integridad de los datos.

- Seguridad de acceso y auditoría.

- Respaldo y recuperación.

- Acceso a través de lenguajes de programación estándar.

### 2.10.2. SQLITE

SQLite es un sistema de gestión de bases de datos que se diferencia de bases de datos convencionales como MySQL u Oracle en que esta lee y escribe archivos binarios independientes (en un estilo similar al de las BBDD de Microsoft Access) [30].

A continuación describiremos sus características [31] más importantes:

- Escrita en lenguaje C y de dominio público (es gratuito para su uso para cualquier propósito, comercial o privado).

- Tiene una pequeña memoria y con una sola biblioteca, es suficiente el acceso a base de datos. Se maneja desde un único archivo de apenas 500KB (en su versión 3, permite bases de datos de hasta 2 Terabytes de tamaño).

- Las bases de datos se guardan en un fichero con extensión ".db".

- Es mucho más rápido que otras BBDD y multiplataforma.

Ventajas [32]:

- Cero configuración (bastará con bajarse "la shell de comandos" de SQLite para empezar a trabajar).

- Multiplataforma y multilenguaje.

- Único archivo de datos.

- Registros de longitud variable.

- Acceso muy rápido.

- Seguridad de datos (Al no haber gestión de usuarios, se basa en los permisos de ficheros establecidos en GNU/Linux).

## 2.10.3. SQLITE EN ANDROID

SQLite se encuentra embebido en los dispositivos Android, por lo que no es necesaria su descarga, configuración y administración. Tampoco es necesario incluir librerías adicionales para trabajar con una base de datos SQLite ya que todas las librerías necesarias pertenecen al kit de desarrollo de aplicaciones en Android.

Utiliza el lenguaje SQL para las consultas (SELECT), manipulación de datos (INSERT, DELETE, etc.), y de definición de datos (CREATE TABLE, etc), soportando los tipos de datos TEXT (String en Java), INTEGER (Integer en Java) y REAL (Double en Java). Si se utiliza cualquier otro tipo de dato, se convierten de manera automática a estos tres tipos de datos.

Para trabajar con SQLite en Android, deberemos usar el paquete android.database, más concretamente android.database.sqlite.

### 2.10.3.1. Paquete android.database.sqlite

Como muestra la tabla 12, la versión de SQLite está relacionada con la versión de la API de Android.

| API de Android | Versión SQLite |
|---|---|
| **API 24** | 3.9 |
| **API 21** | 3.8 |
| **API 11** | 3.7 |
| **API 8** | 3.6 |
| **API 3** | 3.5 |
| **API 1** | 3.4 |

*Tabla 12 – Versiones API y SQLite*

Para crear y actualizar una base de datos en Android, se crea una clase propia que herede de la clase SQLiteOpenHelper.

Dado que para crear y actualizar la BBDD es necesario un objeto SQLiteDatabase, será necesario utilizar métodos de esta clase para su correcta manipulación. La tabla 13 describe los métodos necesarios para este manejo.

| Método | Definición |
|---|---|
| **create(SQLiteDatabase.CursorFactory factory)** | Crea una base de datos con memoria disponible |
| **delete(String table, String whereClause, String[] whereArgs)** | Método usado para borrar filas de una base de datos. |

| | |
|---|---|
| **execSQL(String sql)** | Ejecuta una sentencia SQL que no sea un SELECT o cualquier otra sentencia que devuelva datos. |
| **execSQL(String sql, Object[] bindArgs)** | Ejecuta una sentencia SQL que no sea un SELECT, INSERT, UPDATE o DELETE. |
| **getVersion()** | Devuelve la versión de la base de datos. |
| **insert(String table, String nullColumnHack, ContentValues values)** | Método usado para insertar una fila en una base de datos. |
| **query()** | Consulta una tabla pasada como parámetro y devuelve un Cursor sobre ese conjunto de datos. |
| **rawQuery()** | Ejecuta un SQL pasado por parámetro y devuelve un Cursor sobre ese conjunto de datos. |
| **setVersion(int version)** | Asigna la versión a la base de datos. |

*Tabla 13 - Métodos android.database.sqlite*

### 2.10.3.2. Clase android.database.sqlite.SQLiteOpenHelper

Ahora ya conocemos las herramientas necesarias para poder manipular la base de datos, ahora necesitaremos conocer cómo podemos crear y actualizar dicha BBDD. Tal y como muestra la tabla 14, esto se consigue mediante la clase SQLiteOpenHelper y sus métodos “onCreate” y “onUpgrade”.

**MÉTODOS:**

| Método | Definición |
|---|---|
| **close()** | Cierra cualquier BBDD abierta. |
| **getReadableDatabase()** | Crea y/o abre una BBDD. |
| **getWritableDatabase()** | Crea y/o abre una BBDD que se usará posteriormente para escribir o leer en ella. |
| **onCreate()** | Este método es llamado cuando la BBDD es creada por primera vez. |
| **onUpgrade()** | Este método es llamado cuando la BBDD necesita ser actualizada. |

*Tabla 14 - Métodos clase android.database.sqlite.SQLiteOpenHelper*

Pero, ¿cómo voy a poder abrir la BBDD cada vez que tenga información nueva? Muy sencillo:

Antes de nada, lo primero es crear un objeto SQLiteOpenHelper, pasándole como parámetro la referencia a la actividad principal, el nombre de la base de datos, un objeto CursorFactory (normalmente null) y la versión de nuestra BBDD [33]. Con esto, podemos tener tres opciones:

1. La BBDD que queremos abrir no existe: La opción más sencilla, se llamará automáticamente al método onCreate() para crearla y, a continuación, se conectará a ella para abrirla.

2. La BBDD que queremos abrir existe y tiene la misma versión pasada por parámetro: Al haber “acertado” con la BBDD, simplemente se conectará y abrirá.

3. La BBDD que queremos abrir existe pero tiene una versión distinta a la que se ha pasado como parámetro: se llamará automáticamente al método onUpgrade() para actualizar los datos; tras actualizar la BBDD, se conectará a ella para abrirla posteriormente.

# CAPÍTULO 3 – DESCRIPCIÓN DE LA APP

Los siguientes apartados estarán dirigidos enteramente a la propia aplicación; al boceto inicial, la estructura general de la aplicación, hablaremos de la herramienta usada en su programación y, finalmente, en una descripción detallada de los bloques más importantes de los que constaría la aplicación.

## 3.1. BOCETO E IDEA PRINCIPAL

La idea principal del proyecto es que fuera sencillo a la par que interactivo e interesante al usuario. Dado que iba a estar enfocada para un público de edad temprana, debía ser vistosa y agradable a la vista.

Tras unas primeras tutorías con David Griol, se comenzó a juntar las ideas que iban surgiendo para poder completar todas las dudas que acontecían. También se determinó que la aplicación se apoyaría en el sistema de bases de datos SQLite y que sería necesaria la implementación del sistema TTS y Recognizer (reconocimiento de voz).

Así y como indica la figura 18, iba tomando forma la aplicación.

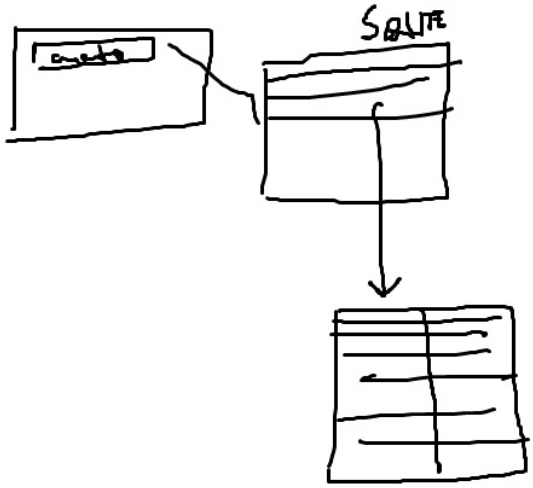



*Figura 18 - Boceto inicial aplicación*

Durante las siguientes semanas, la mayor parte del tiempo se dedicó a obtener una idea más clara de la interfaz (o diseño) de la aplicación. Como muestran las figuras 19-20-21-22, inicialmente se decidió que la aplicación estuviese formada de la siguiente forma.


*Figura 19 - Boceto inicial de la lista de cuentos*


*Figura 20 - Boceto de editar/añadir cuento*

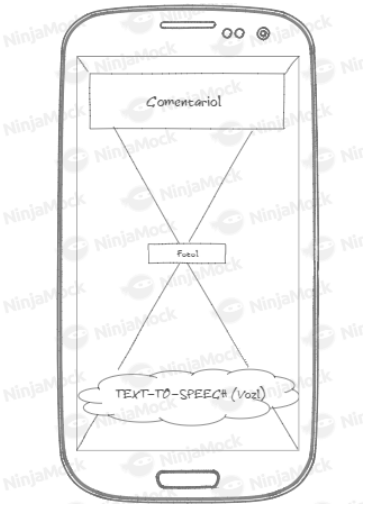



*Figura 21 - Boceto de página genérica del cuento*


*Figura 22 - Boceto de los detalles del cuento*

También mencionar que este diseño previo no es el diseño definitivo ya que, tras ir programando la aplicación surgieron dudas que hicieron volver a replantear la interfaz de la aplicación, originando el diseño que se aportará en la sección 3.3. “Estructura general de la app”.

## 3.2. PROGRAMANDO EN ANDROID STUDIO

Una vez decidido el Sistema Operativo y, tal y como hemos dicho en la sección 2.6 "Por qué Android Studio", se hablará sobre los requerimientos del sistema, los asistentes presentes dentro del propio entorno de programación, y la estructura general de un proyecto en Android Studio.

### 3.2.1. REQUISITOS DEL SISTEMA

Los requerimientos expuestos a continuación son los ofrecidos en la página Android Developer [34] para la última versión de Android Studio (versión 2.3.3) en un ordenador con S.O. Windows:

- Microsoft Windows 7/8/10.

- 3GB RAM mínimo más 1GB para el AVD de Android.

- 2GB de espacio en disco disponible como mínimo, 500MB para el IDE y 1,5GB adicionales para SDK y la imagen del emulador.

- 1280x800 de resolución de pantalla.

- Para el emulador acelerado: Sistema operativo de 64 bits y procesador Intel® compatible con Intel® VT-x, Intel® EM64T (Intel® 64) y la funcionalidad Execute Disable (XD) Bit.

## 3.2.2. ASISTENTES DE ANDROID STUDIO

Dos de las principales características y ventajas de Android Studio, como hemos mencionado en la sección 2.5.1, son que se actualiza constantemente y que posee un emulador que nos proporciona una vista previa de cómo va nuestra aplicación.

Para que estas características puedan llevarse a cabo, Android Studio incorpora dos "asistentes", los cuales son los siguientes:

**- SDK Manager:** Nos informa qué complementos tenemos hasta el momento instalados en nuestra máquina y cuáles no. Se recomienda siempre seleccionar todas las actualizaciones disponibles e instalarlas, debido a que en algún momento puede ser necesaria. El punto en contra de esto es que el tiempo requerido para completar todas las instalaciones será mayor cuantas más opciones seleccionemos.

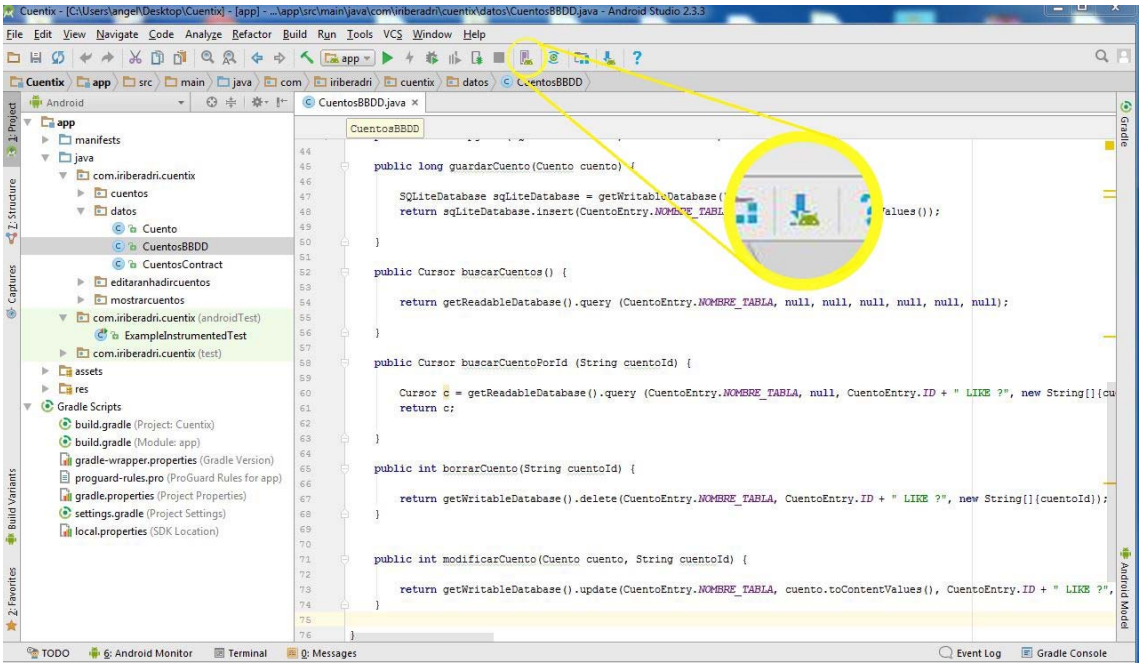


*Figura 23 - Botón SDK Manager*

Al pulsar dicho botón, nos aparecerá un cuadro como la figura 24:

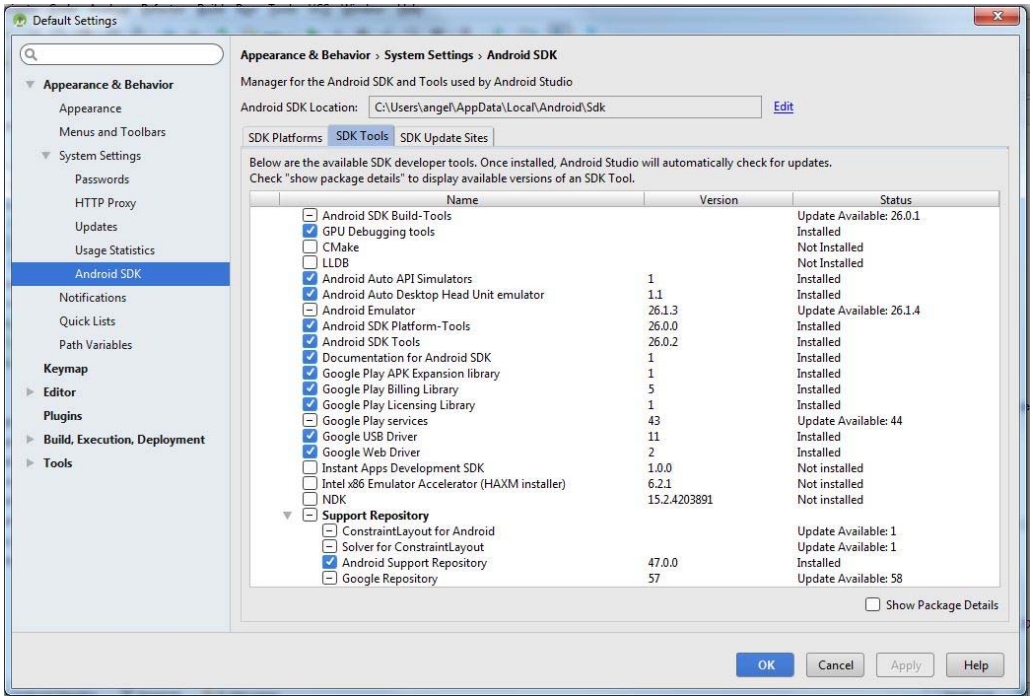



*Figura 24 - Ventana SDK Manager*

**- Android Virtual Device Manager (ADV Manager):** Una AVD es una máquina virtual que ejecuta la plataforma Android seleccionada, para comprobar la funcionalidad de nuestra app. El SDK trae consigo una pequeña aplicación de gestión de AVDs llamada AVD Manager, por lo que la ejecutaremos para crear un nuevo dispositivo virtual.

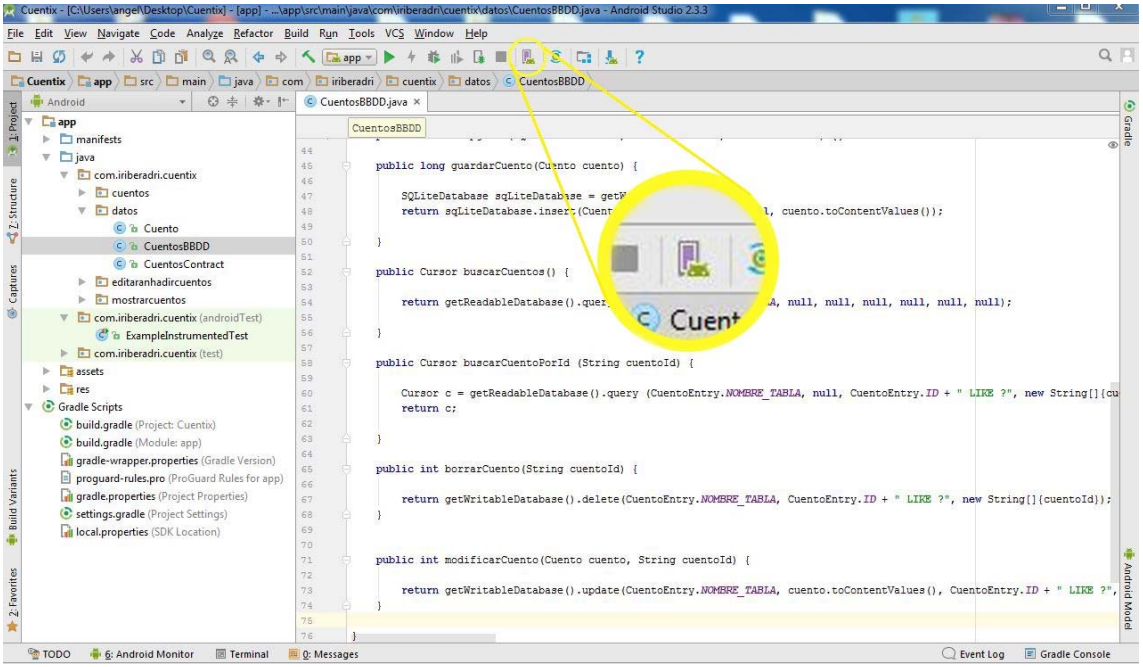



*Figura 25 - Botón AVD Manager*

Al pulsar dicho botón, nos aparecerá otro cuadro como la figura 26:

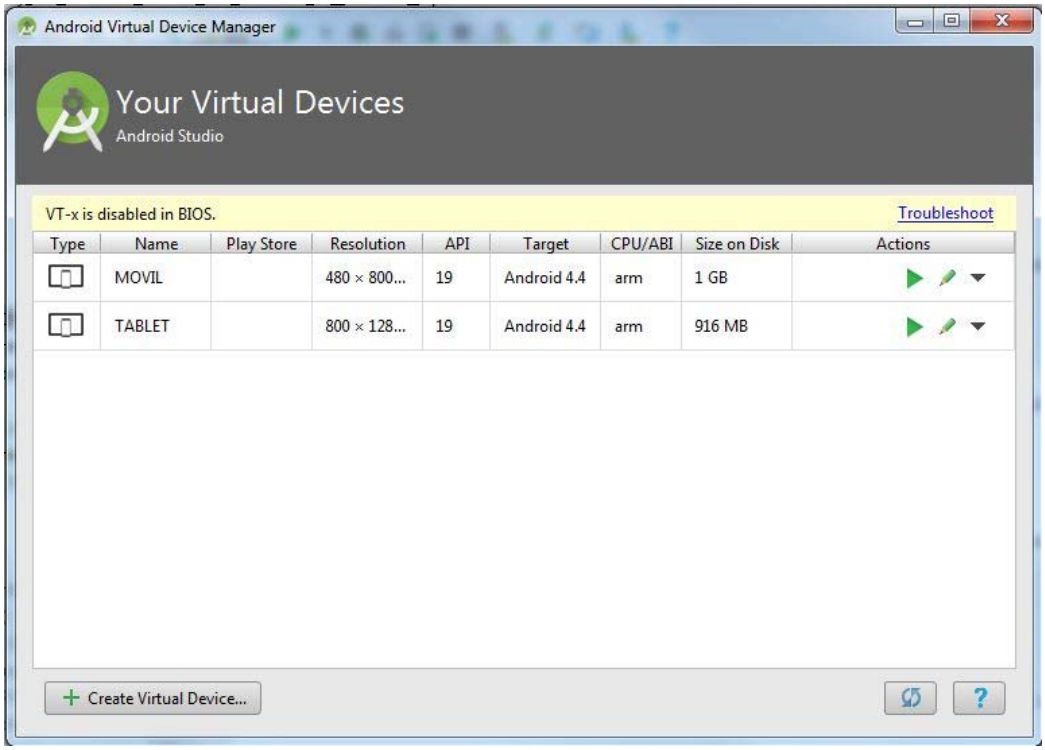



*Figura 26 - Ventana AVD Manager*

## 3.3. ESTRUCTURA GENERAL DE LA APP

En este punto se explicará con detalle de qué trata la aplicación y cuál es su funcionamiento desde el primer momento en el que el usuario selecciona dentro del "saco" de apps en su dispositivo, hasta el momento en el que éste decide cerrarla al terminar con ella.

El esquema general de la aplicación consta de 3 partes tal y como muestra la figura 27 en la que se le pide al usuario que introduzca por teclado una acción para ejecutar el cuento, editar el cuento o crear uno nuevo. En el caso de haber elegido que ejecute el cuento, el usuario tendrá la opción de ir avanzando en la historia de dos modos, tanto por teclado como por reconocimiento de voz.

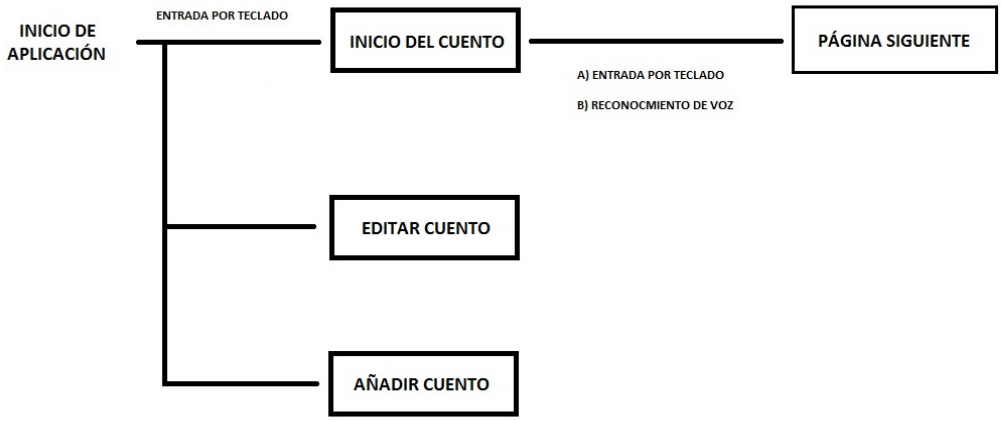



*Figura 27 - Esquema general Cuentix*

Acabamos de ver el esquema general y a grandes rasgos que constaría la aplicación. A continuación explicaremos detalladamente todo el proceso desde conectarse a la app, hasta cerrarla:

### 3.3.1. INICIO DE APLICACIÓN

Este es el punto de partida de todo el proceso, en el que el usuario, tras colocarse en el "saco" de aplicaciones del dispositivo, buscará y seleccionará el icono de "Cuentix", es decir, de nuestra app tal.

La figura 28 presenta las aplicaciones instaladas en el dispositivo y, entre todas ellas, se encuentra la nuestra.


*Figura 28 - Inicio aplicación*

### 3.3.2. SELECCIÓN DE CUENTO

Tras iniciarse la aplicación y aparecer, durante un par de segundos, el icono de la aplicación, aparecerá la pantalla principal en la que se encuentran todos los cuentos instalados en el dispositivo.

En esta ventana tendremos tres opciones de acción:

**1. Crear cuento:** Con esta selección, podremos crear un nuevo cuento desde cero.

**2. Acción sobre cuento instalado:** Seleccionaremos uno de los cuentos de la lista para editarlo o ejecutarlo.

**3. Salir de la aplicación:** Bastará con dar dos veces al botón "atrás" del dispositivo, una para realizar la acción y otra para confirmar que se quiere salir de la app.

Señalar que la primera vez que se inicia la aplicación, o si no se ha creado ningún cuento nuevo, aparecerá el cuento de muestra "Caperucita Roja", como muestra la figura 29.

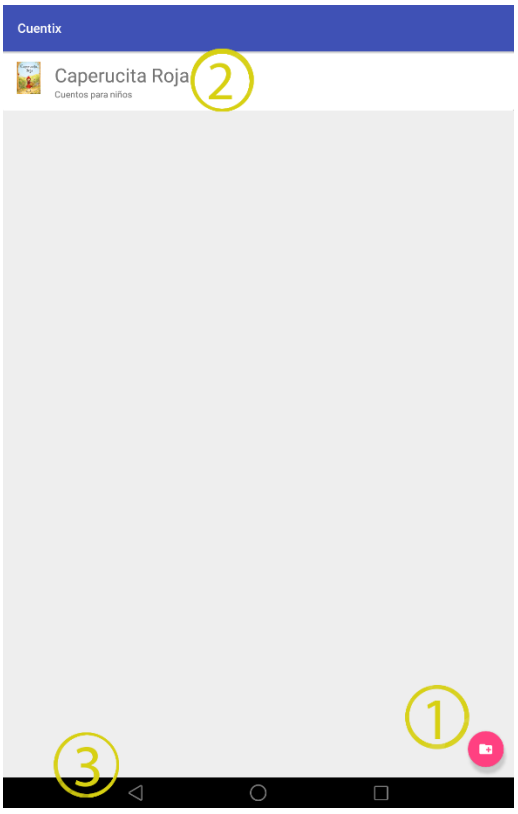



*Figura 29 - Lista de cuentos*

#### 3.3.2.1. Crear cuento

Esta opción nos da la oportunidad de poder crear nuestro propio cuento con sus propias imágenes y texto. Además, la ventana a la que se nos direcciona tras presionar el botón de añadir, será muy parecido a la de "Detalles" del cuento, como veremos más abajo.

A modo de ejemplo, usaremos la figura 30 como estructura para la creación de un cuento.

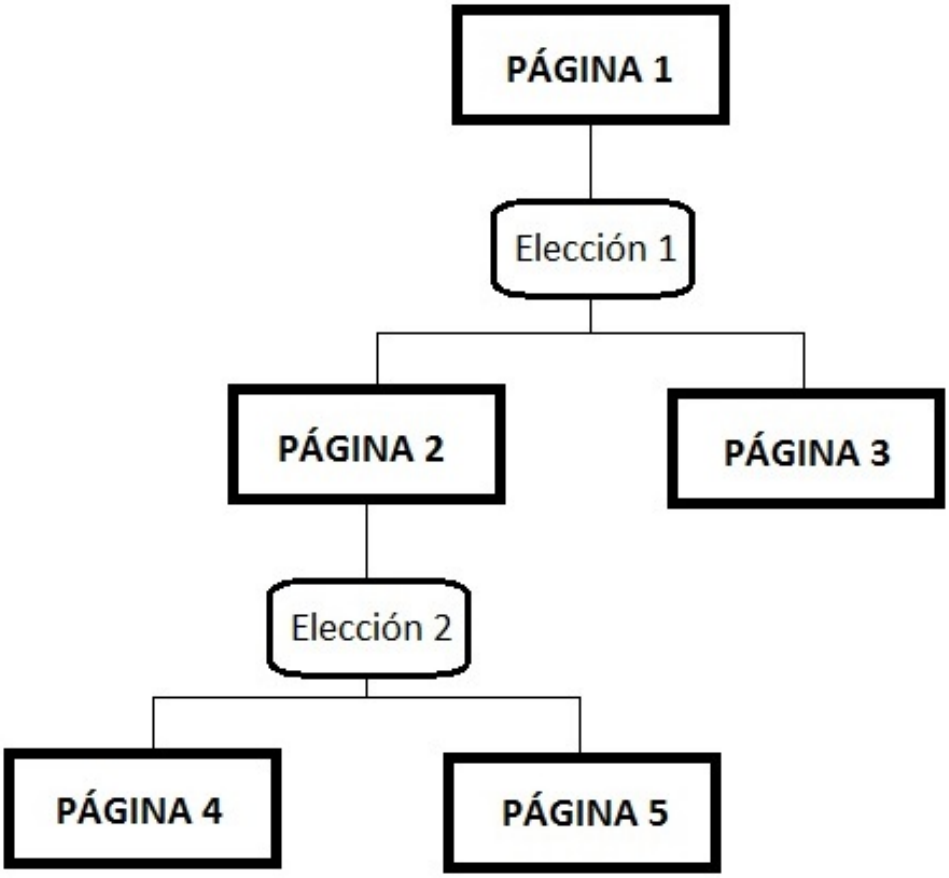



*Figura 30 - Estructura general de un cuento*

Una vez seleccionado el botón, nos aparecerá una ventana emergente indicándonos que, antes de realizar cualquier acción sobre las páginas, es necesario guardar el cuento. Tras aceptar esta consideración, nos aparecerá la ventana descrita con todos los campos en blanco, como indica la figura 31.


*Figura 31 - Crear un cuento*

Ahora comenzaremos con la creación del cuento. Primeramente, añadiremos la imagen de portada y el título del cuento. Una vez seleccionados, presionaremos el botón “Guardar cuento”. Si no se ha completado alguno de estos dos campos, aparecerá un mensaje indicándonos que se deben introducir ambos, como muestra la figura 32.

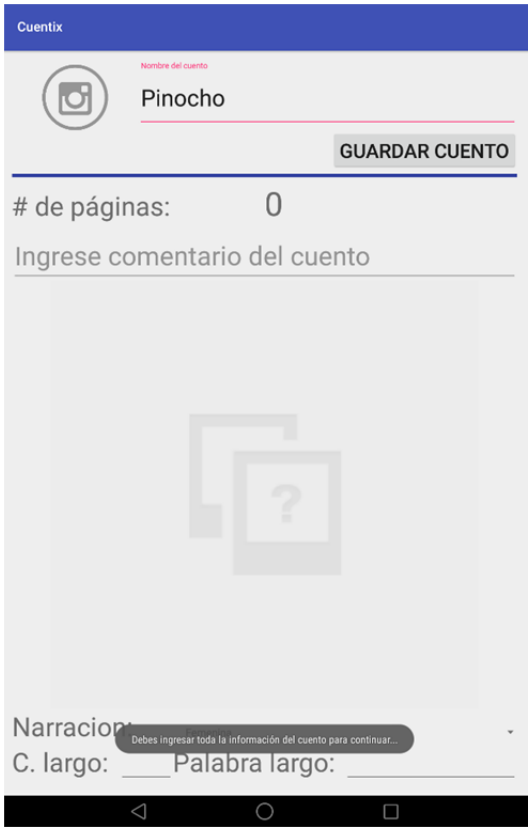



*Figura 32 - Error al crear título y portada*

Por el contrario, si se han insertado los dos campos y presionamos el botón, nos indicará que el cuento se ha guardado correctamente, como muestra la figura 33.

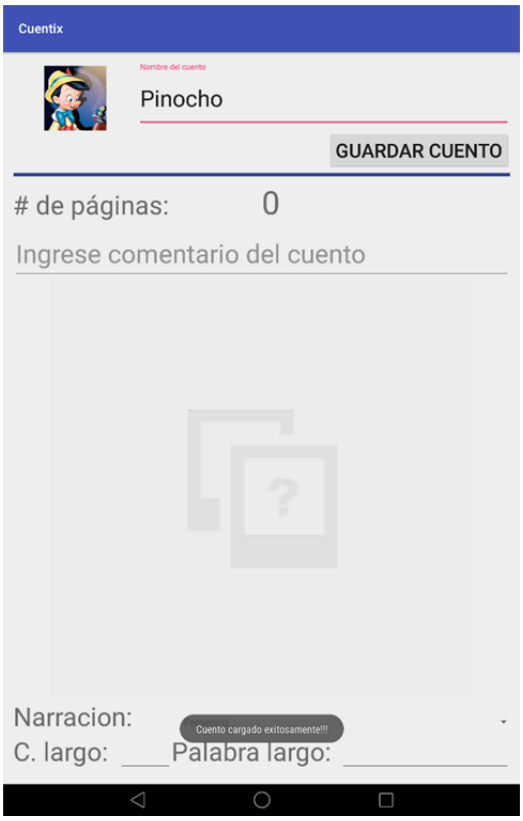



*Figura 33 - Éxito al guardar título y portada*

Una vez guardado el título y la portada, comenzaremos con la creación de las páginas. Si volvemos a guiarnos de la figura 32, estos son los puntos a la hora de crear una página:

1. Incluiremos el comentario de la página, que será usado por el motor de síntesis de voz para hacer que la aplicación “cuente” dicho texto.

2. Incluiremos la imagen de la página

3. Seleccionaremos, mediante un spinner, el género de la voz que queremos usar en nuestra página. Habrá dos opciones: Masculino y Femenino.

4. Ahora es el momento de seleccionar los caminos. Para esto, pueden darse dos opciones: que sea una página con una elección que derivará en dos páginas (es el caso de las páginas 1 y 2) o que sea una página de fin de cuento (caso de las páginas 3,4 y 5). Vamos a ahondar estas dos opciones:

- **Páginas derivadas:** Para este caso, deberemos seleccionar las páginas tanto para camino largo como para camino corto y las palabras que serán necesarias para que el reconocimiento de voz compare y elija uno u otro camino, además de deseleccionar "Final del cuento". Hay que indicar que, si nos encontramos en la primera página, el camino largo corresponderá a la página 2 y el camino corto al camino 3.

- **Páginas finales:** En este otro caso, las páginas y las palabras para camino corto y camino largo no importan, es decir, se pueden poner cualquier página. No habrá ningún problema de dirección dado que deberemos seleccionar "Final del cuento"; con esto, indicamos que no habrá ninguna página "colgando de esta".

5. Si ya hemos terminado con la página pero queremos seguir creando más, deberemos seleccionar el botón "Guardar página" y automáticamente seguiremos con la siguiente hoja por orden numérico.

6. Una vez finalizada la última página de nuestro cuento, deberemos seleccionar "Terminar". Se nos redirigirá a la lista de cuentos, añadido nuestro cuento creado.

Tal y como muestra la figura 34, el cuento se ha insertado en la lista de cuentos instalados

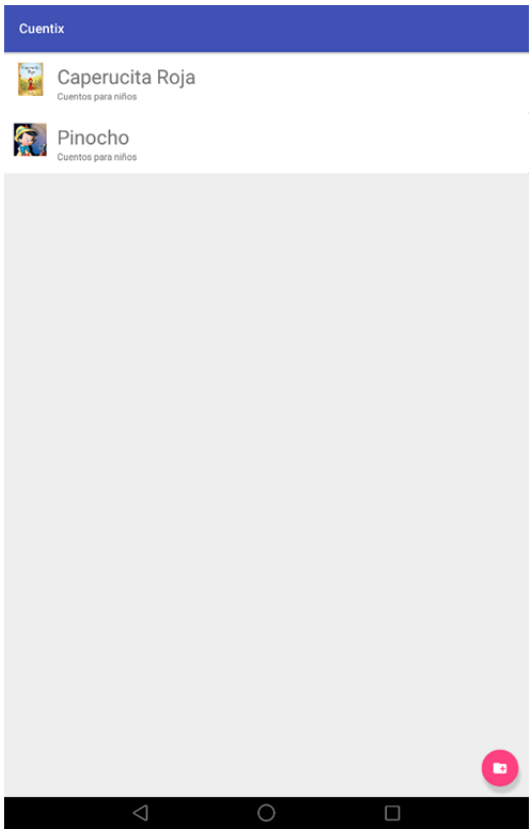



*Figura 34 - Lista de cuentos con cuento nuevo*

### 3.3.2.2. ACCIÓN SOBRE CUENTO INSTALADO

Si presionamos el cuento elegido, en este caso “Caperucita Roja”, nos aparecerá un cuadro con dos opciones, tal y como muestra la figura 35:

**- Detalles:** En esta sección podremos ver las características de nuestro cuento (título, portada, páginas, etc.) y editarlas en el caso de querer sustituir algún aspecto de nuestro cuento.

**- Leer cuento:** Si presionamos, la aplicación comenzará el cuento.

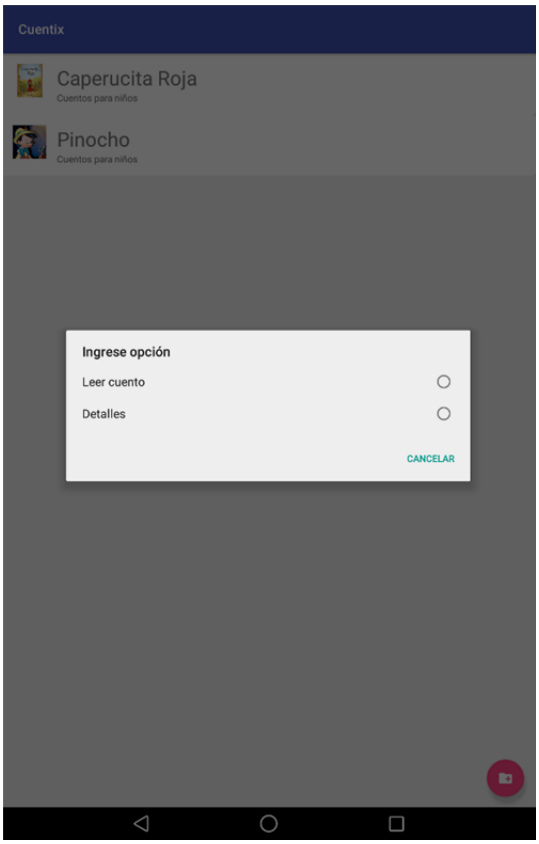



*Figura 35 - Opciones tras acción sobre cuento*

A continuación explicaremos con más detalles cada una de las dos opciones expuestas anteriormente:

## Detalles

En esta parte se encuentran todas las características de nuestro cuento, desde el título y la portada, hasta las elecciones de cada pantalla y la voz del narrador.


*Figura 36 - Ventana Detalles del cuento*

Como podemos comprobar en la figura 36, en la pantalla que se nos muestra tras seleccionar esta opción, aparecerá:

**- Nombre del cuento:** Se trata del título del cuento. Se podrá cambiar seleccionando el texto y escribiendo el título nuevo.

**- Portada del cuento:** Es la imagen actual del cuento que se utilizará en la lista de cuentos instalados.

**- Cambiar imagen:** Con este botón podremos cambiar la imagen de la portada. Se abrirá a continuación un cuadro de selección de carpetas de imágenes.

**- Actualizar cuento:** Tras realizar algún cambio tanto en el título como en la imagen de portada, será necesario guardar estos datos del cuento.

**- Botón editar:** Es necesario pulsarlo antes de cambiar cualquier característica del cuento.

**- Botón borrar:** Con este botón borraremos el cuento por completo. Tras realizar esta acción, volveremos a la ventana con la lista de cuentos, pero sin el cuento borrado.

**- # de página:** En esta parte de la sección se encontrarán todas las características de la página del cuento, además de los botones para realizar acciones tales como pasar página y ver las siguientes, actualizar, terminar, tal y como muestra la figura 37. Señalar, también, que estos campos son comunes para todas las páginas del cuento.


*Figura 37 - Detalles página de un cuento*

**- Imagen de la página:** En este campo se mostrará la imagen actual que se está utilizando como "portada" de la página en cuestión.

**- Voz:** Se trata de la voz que se usará para el narrador, es decir, la voz con la que la aplicación hablará al usuario.

**- Camino largo:** Se indica la página por la que continuará el cuento tras elegir este camino. Si nos fijamos en la figura 30 de la sección "CREAR CUENTO", será el camino entre 1 y 2 (o entre 3 y 4).

**- Palabra largo:** Será la palabra que tendrá que indicar oralmente el usuario para elegir el camino largo.

**- Camino corto:** Será la palabra que tendrá que indicar oralmente el usuario para elegir el camino corto.

**- Palabra corto:** Se indica la página por la que continuará el cuento tras elegir este camino. Si nos fijamos en la figura 30, será el camino entre 1 y 3 (o entre 3 y 5).

**-Final de cuento:** Esta opción estará marcada en el caso de que la página no conlleve ninguna elección o ninguna página a continuación.

**- Página atrás:** Botón que nos hará regresar a la página anterior del cuento.

**- Página siguiente:** Botón que nos permitirá ver la siguiente página del cuento.

**- Actualizar página:** Con este botón, guardaremos los datos sustituidos de la página en cuestión.

**- Terminar:** Concluiremos con la edición del cuento y regresaremos a la ventana con la lista de cuentos instalados.

### Leer cuento

Con esta selección ejecutaremos el cuento. Nos aparecerá la primera imagen, con el texto en la parte superior y 4 botones, tal y como se muestra en la figura 38:


*Figura 38 - Ejecución del cuento*

**- Escuchar:** Si presionamos este botón, podremos escuchar el texto de la parte superior de la imagen con la voz que hayamos definido, gracias al motor de síntesis de voz configurado, como explicamos en la sección 2.8 “TTS en Android”.

**- Micrófono:** Se le pedirá al usuario que conteste la pregunta realizada por el narrador. Tras el reconocimiento de voz, explicado en la sección 2.9 “Reconocimiento de Voz”, la aplicación hará la comprobación con “Palabra Largo” y “Palabra Corto”. Se pueden dar tres opciones:

- Si coincide con palabra largo, entonces irá a la página configurada en “Camino largo”.

- Si coincide con palabra corto, entonces irá a la página configurada en “Camino corto”.

- Si no coincide con ninguna de las dos, no realizará ninguna acción y será necesario volver a darle al botón y decir una opción.

**- Botón siguiente:** Podremos, además de oralmente, seleccionar por qué camino queremos ir presionando este botón. La figura 39 muestra la ventana emergente tras presionarlo.

**- Botón anterior:** Mediante este botón podremos volver a la página anterior del cuento solamente si la página conlleva una elección de camino.

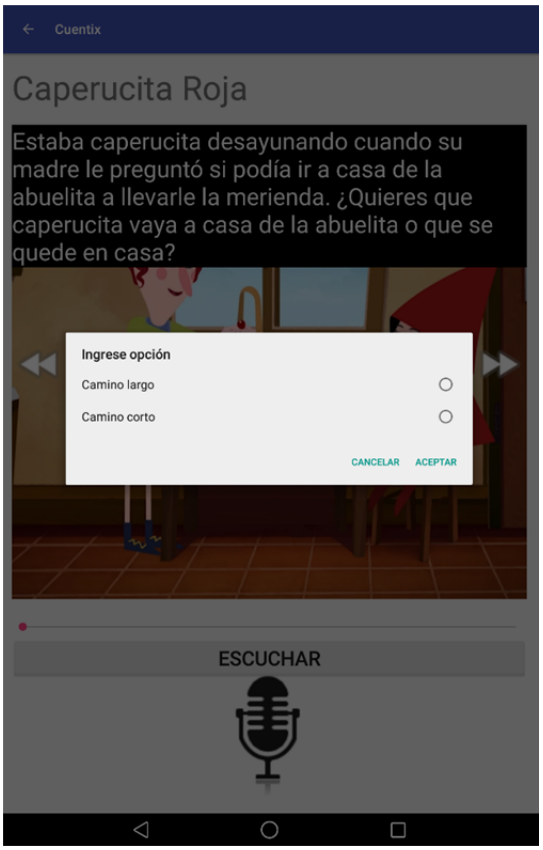



*Figura 39 – Elección de camino entre páginas mediante teclado*

## 3.4. APLICACIÓN DIRIGIDA AL DESARROLLADOR

Para concluir el capítulo, se expondrá el desarrollo de la aplicación, pero desde un punto de vista orientado al desarrollador, haciendo hincapié en aspectos importantes y en las partes que he tenido más problemas a la hora de realizar, tales como inserción de imágenes, manejo de la base de datos con información o la ejecución de los cuentos.

En primer lugar, la aplicación consta de 6 actividades que ayudarán a desenvolver toda la funcionalidad de la misma. La aplicación, al iniciarse, aparecerá un SplashScreen mostrando una imagen con el logo de Cuentix durante un segundo.

```
public class SplashScreenActivity extends AppCompatActivity {
    private static int SPLASH_TIME_OUT = 1000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);

        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        getSupportActionBar().hide();
        new Handler().postDelayed(new Runnable() {

            /*
             * Muestra una SplashScreen con un temporizador. Será
             * útil a la hora de mostrar el logo de la empresa.
             */

            @Override
            public void run() {
                // Una vez que concluya el temporizador, la aplicación se ejecutará.
                Intent i = new Intent(SplashScreenActivity.this, MainActivity.class);
                startActivity(i);

                // Se cierra la actividad.
                finish();
            }
        }, SPLASH_TIME_OUT);
    }
}
```

*Figura 40 - Activity SplashScreen*

La figura 40 detalla cómo se realiza esta acción. Como podemos comprobar, esta clase posee un método onCreate() que es propio de la actividad. Indicaremos, predeterminadamente, que el SplashScreen se muestre en Portrait, ocultando el ActionBar mediante el método getSupportActionBar().hide(). Al usar este método, debemos crear otro que nos permita ejecutar dicho SplashScreen durante un cierto intervalo de tiempo y luego poder iniciar la actividad principal (MainActivity).

En esta actividad, lo que veremos será un RecyclerView con un CardView, con todos los cuentos que se agregarán. En primer lugar, se declararán las variables correspondientes para obtener su instancia y las variables que se necesitarán.

Se utilizarán dos tablas como bases de datos para guardar los cuentos y poder ejecutarlos cada vez que se ejecute la aplicación.

Tal y como muestra la figura 41, la disposición de las dos tablas será la siguiente:

CUENTO

| ID_CUENTO | TITULO | PORTADA |
|---|---|---|
| 1 | Caperucita Roja | Portada_Cap |
| 2 | Pinocho | Portada_Pin |
| | | |

PAGINAS

| ID_CUENTO | ID_PAG | COMENTARIO | VOZ | FOTO | CAMINO LARGO | CAMINO CORTO | PALABRA LARGO | PALABRA CORTO | FINAL CUENTO |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Comentario1 | Voz1 | Foto1 | 2 | 3 | Largo1 | Corto1 | 0 |
| 1 | 2 | Comentario2 | Voz2 | Foto2 | 0 | 0 | - | - | 1 |
| 1 | 3 | Comentario3 | Voz3 | Foto3 | 0 | 0 | - | - | 1 |
| | | | | | | | | | |

*Figura 41 - Tablas de BBDD en Cuentix*

En primer lugar debe de crearse la Base de Datos, si es que no ha sido creada. Para esto, usaremos la siguiente línea de código:

```
db = this.openOrCreateDatabase("Cuentix.db",Context.MODE_PRIVATE,null);
```

*Figura 42 - Creación de BBDD de Cuentix*

- Si es la primera vez que se ejecuta la aplicación, la Base de Datos se creará.

- Si la aplicación se ha ejecutado más de una vez, se abrirá la Base de Datos.

Tras esta acción, se crearán las tablas que se usarán en la aplicación. Como hemos visto en la figura 53, las tablas se llamarán “Cuento” y “Pagina”. Las dos tablas están relacionadas entre sí; es decir, si la tabla “Cuento” no existe, no pueden existir páginas para ese cuento.

Suponiendo que es la primera vez que se ejecuta la aplicación y que ya se han creado las tablas, la siguiente acción será ingresar datos en dichas tablas. Las figuras 43 y 44 reflejan esta acción:

```
File f = new File(getCacheDir()+"/caperucita.jpg");
if(!f.exists()){
    try{
        InputStream is = getAssets().open("caperucita.jpg");
        int size = is.available();
        byte[] buffer = new byte[size];
        is.read(buffer);
        is.close();

        FileOutputStream fos = new FileOutputStream(f);
        fos.write(buffer);
        //Agrego el nombre, la imagen al cuento, y los inserto en la Base de Datos
        ContentValues values = new ContentValues();
        values.put("titulo","Caperucita Roja");
        values.put("a",buffer);
        db.insert("cuento",null,values);

        is = getAssets().open("foto1.png");
        size = is.available();
        buffer = new byte[size];
        is.read(buffer);
        is.close();
```

*Figura 43 - Inserción de info en tabla Cuento*

```
is = getAssets().open("foto1.png");
size = is.available();
buffer = new byte[size];
is.read(buffer);
is.close();

fos = new FileOutputStream(f);
fos.write(buffer);
ContentValues values1 = new ContentValues();
values1.put("id_cuento",1);
values1.put("pagina",1);
values1.put("comentario","Estaba caperucita desayunando cuando su madre le preguntó
values1.put("voz","Femenina");
values1.put("foto",buffer);
values1.put("a",2);
values1.put("b",3);
values1.put("palabra_a","sí");
values1.put("palabra_b","no");
values1.put("final",false);
db.insert("pagina",null,values1);
```

*Figura 44 - Inserción de info en tabla Pagina*

Las imágenes se buscarán en el directorio Assets, convirtiéndose cada una a un arreglo de bit para poder guardarlas en la Base de Datos con un campo Blob, permitiéndonos guardar toda la información de la imagen. Además, con este campo, aunque la imagen se borre, no se pierde información y podremos seguir cargándola en el cuento. Tras esto, se insertan todos los campos en la tabla y se llama al método db.insert para incluirlos en la Base de Datos.

A continuación, se buscan los cuentos que se encuentran dentro de la Base de Datos para poder mostrarlos en la actividad principal (la ventana de lista de cuentos), tal y como muestra la figura 45:

```
//Se obtienen todos los cuentos de la base de datos
cuentoList = new ArrayList<>();
Cursor c = db.rawQuery("Select * from cuento",null);
if(c != null){
    if(c.moveToFirst()){
        do{
            Cuento cuento = new Cuento();
            cuento.setName(c.getString(1));
            cuento.setStory("Cuentos para niños");
            cuento.setPhoto(c.getBlob(2));
            cuentoList.add(cuento);
        }while (c.moveToNext());
    }
}
```

*Figura 45 - Obtención de cuentos de la BBDD*

Como vemos, se ejecuta el método rawQuery() y se declara un nuevo objeto Cuento, donde se guardará toda la información del cuento que aparece en la Base de Datos.

Por último, y como muestra la figura 46, se carga toda la información en el RecyclerView descrito anteriormente para ser mostrada en la pantalla:

```
//Cargamos, en el RecyclerView, los cuentos almacenados en la BBDD
recyclerView = (RecyclerView) findViewById(R.id.rv);
recyclerView.setHasFixedSize(true);
layoutManager = new LinearLayoutManager(this);
recyclerView.setLayoutManager(layoutManager);
recyclerView.setAdapter(new CuentoAdapter(this,cuentoList,(view, position)  -> {
        showMsg(context,true,cuentoList.get(position).getName());
}));
mBtnAñadir = (FloatingActionButton) findViewById(R.id.fab);

mBtnAñadir.setOnClickListener((view)  -> { MostrarAgregar(); });
```

*Figura 46 - Insertar cuentos desde BBDD en RecyclerView*

A continuación, veremos cómo se ingresarán todos los datos de un cuento nuevo y sus páginas. Para ello, se abrirá la Base de Datos.

```
//Se obtiene la imagen de la galeria, se convierte en un arreglo de bit  y se inserta
cargar.setOnClickListener((view)  -> {
        boolean a = validateName();
        if(imageUri !=null && a){
            String[] filePathColumn = {MediaStore.Images.Media.DATA};
            Cursor cursor = getContentResolver().query(imageUri, filePathColumn, null, null, null);
            if(cursor.moveToFirst()){
                int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
                String yourRealPath = cursor.getString(columnIndex);
                Log.e("RealPath",yourRealPath);
                try {
                    FileInputStream fis = new FileInputStream(yourRealPath);
                    byte[] image = new byte[fis.available()];
                    fis.read(image);
                    cuento = new Cuento(nombreCuento.getText().toString(),"Cuentos para niños",image);
                    ContentValues values = new ContentValues();
                    values.put("titulo",nombreCuento.getText().toString());
                    values.put("a",image);
                    db.insert("cuento",null,values);
                    Toast.makeText(context,"Cuento cargado exitosamente!!!",Toast.LENGTH_SHORT).show();
                    fis.close();
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } else {
                //buuuhh, el cursor no tiene filas ...
            }
            cursor.close();
```

*Figura 47 - Botón de Cargar Cuento*

Uno de los métodos más importantes es el de abrir la galería para incluir las imágenes, ya que se abrirán dos galerías, una para la foto de portada y otra para la de la propia página. Tal y como mencionamos al principio de este punto, es necesario ingresar los datos del cuento (título y portada) para después poder ingresar los datos de las páginas.

Como se muestra en la figura 47, aquí se ve si se ha cargado una imagen para el cuento. Si se ha cargado, se convierte en un arreglo de bit y se carga en la Base de Datos con el resto de los datos. En el caso de no cargarse correctamente, aparecerá una ventana emergente indicando que es necesario cargar la imagen antes de guardar el cuento.

Indicar, también, que el caso del botón destinado a cargar las páginas es igual que el anterior, con la diferencia que hay más campos que ingresar la base de datos; sin embargo, la validación de cargar la imagen de la página antes que guardar la página del cuento sigue siendo la misma.

En la siguiente figura 48 se muestra cómo se cargaría la página del cuento, tras incluir los datos de la página en la Base de Datos y pasaría a la siguiente página directamente tras pulsar “Guardar Página”.

```
//Insertamos los datos de la nueva página  y los incluimos en la BBDD
sig_pag.setOnClickListener((view)  -> {
        boolean aa = validateComent();
        boolean bb = validateA();
        boolean cc = validateB();
        if(imageUri2 != null && aa && bb && cc){
            //Obtener cuento
            Cursor c = db.rawQuery("Select * from cuento where titulo='"+nombreC+"'",null);
            if(c != null){
                if(c.moveToFirst()){
                    do{
                        id = c.getInt(0);
                        Log.e("id:",Integer.toString(id));
                    }while (c.moveToNext());
                }
            }
            String[] filePathColumn = {MediaStore.Images.Media.DATA};
            Cursor cursor = getContentResolver().query(imageUri2, filePathColumn, null, null, null);
            if(cursor.moveToFirst()) {
                int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
                String yourRealPath = cursor.getString(columnIndex);
                Log.e("RealPath", yourRealPath);
                try {
                    //Insertar paginas
                    FileInputStream fis = new FileInputStream(yourRealPath);
                    byte[] image = new byte[fis.available()];
                    fis.read(image);
                    Pagina pagina = new Pagina(id,numeroPag,contenido.getText().toString(),spinner.getSelectedItem()
                    ContentValues values = new ContentValues();
                    values.put("id_cuento",pagina.getId_cuento());
                    values.put("pagina", pagina.getNumeroPag());
```

*Figura 48 - Botón Guardar Pagina nueva*

Tras ver cómo se insertan cuentos nuevos en la aplicación, vamos a ver cómo se pueden ver y editar los detalles de cada cuento que ya ha sido guardado.

Primeramente, como siempre, abriremos la Base de Datos y obtendremos el dato del cuento que ha sido seleccionado para mostrar en detalles. Esta acción se refleja en la figura 49:

```
//Abrimos la base de datos
db = this.openOrCreateDatabase("Cuentix.db", Context.MODE_PRIVATE,null);
Intent intent = getIntent();
name = intent.getStringExtra("Name");

//Obtenermos los datos del cuento abierto
cuento = new Cuento();
Cursor c = db.rawQuery("Select * from cuento where titulo='"+name+"'",null);
if(c!=null){
    if(c.moveToFirst()){
        do{
            id_cuento = c.getInt(0);
            cuento = new Cuento();
            cuento.setName(c.getString(1));
            cuento.setStory("Cuentos para niños");
            cuento.setPhoto(c.getBlob(2));
        } while (c.moveToNext());
    }
}
```

*Figura 49 - Abrir BBDD y obtener datos del cuento guardado*

Tras obtener los datos del cuento guardado, los ingresaremos en los EditText para que, más tarde, podamos editarlos, tal y como vemos en la figura 50:

```
//Ingresar datos del cuento en EditText para editarlos
Bitmap bitmap = BitmapFactory.decodeByteArray(cuento.getPhoto(),0,cuento.getPhoto().length);
collapImage.setImageBitmap(bitmap);
newName.setText(cuento.getName());
newName.setFocusable(false);
```

*Figura 50 - Insertar datos del cuento en EditText para editar*

Al igual que hemos obtenido los datos del cuento y los hemos ingresado en los EditText, haremos con las páginas del cuento, como indican las figuras 51 y 52:

```
//Obtenermos los datos de las páginas del cuento
list = new ArrayList<>();
c = db.rawQuery("Select * from pagina where id_cuento='"+id_cuento+"'",null);
if(c!=null){
    if(c.moveToFirst()){
        do{
            Pagina pagina = new Pagina();
            pagina.setId_cuento(c.getInt(1));
            pagina.setNumeroPag(c.getInt(2));
            pagina.setComentario(c.getString(3));
            pagina.setVoz(c.getString(4));
            pagina.setFoto(c.getBlob(5));
            pagina.setA(c.getInt(6));
            pagina.setB(c.getInt(7));
            pagina.setPalabra_a(c.getString(8));
            pagina.setPalabra_b(c.getString(9));
            pagina.setCheck(c.getInt(10) > 0);
            Log.e("Id_cuento",Integer.toString(pagina.getId_cuento()));
            list.add(pagina);
        }while (c.moveToNext());
    }
}
```

*Figura 51 - Obtener datos de la página en la BBDD*

```
//Ingresamos los datos de la página seleccionada en EditText para editarlos
if(list.size() != 0){
    i = 0;
    //i = 1;
    numPages.setText(Integer.toString(list.get(0).getNumeroPag()));
    newComent.setText(list.get(0).getComentario());
    newComent.setFocusable(false);
    Bitmap bitmap1 = BitmapFactory.decodeByteArray(list.get(0).getFoto(),0,list.get(0).getFoto().length);
    newComentImage.setImageBitmap(bitmap1);
    final ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
            R.array.Voz, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    // Apply the adapter to the spinner
    newSpinner.setAdapter(adapter);
    newSpinner.setFocusable(false);
    newA.setText(Integer.toString(list.get(0).getA()));
    newA.setFocusable(false);
    newB.setText(Integer.toString(list.get(0).getB()));
    newB.setFocusable(false);
    newPA.setText(list.get(0).getPalabra_a());
    newPA.setFocusable(false);
    newPB.setText(list.get(0).getPalabra_b());
    newPB.setFocusable(false);
    newCheck.setChecked(list.get(0).isCheck());
    newCheck.setFocusable(false);
```

*Figura 52 - Insertar datos de la página en EditText para editar*

En el supuesto caso que el cuento no tuviera páginas, se colocarían todos los campos como si no tuvieran nada, exactamente como indica la figura 53:

```
}else{
    numPages.setText(Integer.toString(0));
    newComent.setText("");
    newComent.setFocusable(false);
    newComentImage.setImageResource(R.drawable.noimage);
    final ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
            R.array.Voz, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    // Aquí se aplica el Adapter al Spinner
    newSpinner.setAdapter(adapter);
    newSpinner.setFocusable(false);
    newA.setText("");
    newA.setFocusable(false);
    newB.setText("");
    newB.setFocusable(false);
    newPA.setText("");
    newPA.setFocusable(false);
    newPB.setText("");
    newPB.setFocusable(false);
    newCheck.setChecked(false);
    newCheck.setFocusable(false);
}
```

*Figura 53 - Insertar página vacía en EditText*

Para poder recorrer las distintas páginas del cuento, se crea un contador i que ayudará a realizar este recorrido. Las figuras 54 y 55 muestran cómo se han implementado los botones para ir a la siguiente página y a la anterior:

```
//Botón para ir a la página siguiente
nextPage.setOnClickListener((view)  -> {
        if(i < list.size()){
            i++;
            //Colocamos la información de la página siguiente
            numPages.setText(Integer.toString(list.get(i).getNumeroPag()));
            newComent.setText(list.get(i).getComentario());
            Bitmap bitmap1 = BitmapFactory.decodeByteArray(list.get(i).getFoto(),0,list.get(i).getFoto().length);
            newComentImage.setImageBitmap(bitmap1);
            comen = list.get(i).getComentario();
            newA.setText(Integer.toString(list.get(i).getA()));
            newB.setText(Integer.toString(list.get(i).getB()));
            newPA.setText(list.get(i).getPalabra_a());
            newPB.setText(list.get(i).getPalabra_b());
            newCheck.setChecked(list.get(i).isCheck());
        }else{
            Toast.makeText(context,"El cuento ya no tiene más páginas",Toast.LENGTH_SHORT).show();
        }
});
```

*Figura 54 - Botón para ver detalles de la página siguiente*

```
//Botón para volver a la página anterior
backPage.setOnClickListener((view) -> {
        if(i == 0){
            Toast.makeText(context, "El cuento no tiene más páginas para atrás", Toast.LENGTH_SHORT).show();
        }else{
            //Colocamos la información de la página anterior
            i--;
            numPages.setText(Integer.toString(list.get(i).getNumeroPag()));
            newComent.setText(list.get(i).getComentario());
            Bitmap bitmap = BitmapFactory.decodeByteArray(list.get(i).getFoto(),0,list.get(i).getFoto().length);
            newComentImage.setImageBitmap(bitmap);
            comen = list.get(i).getComentario();
            newA.setText(Integer.toString(list.get(i).getA()));
            newB.setText(Integer.toString(list.get(i).getB()));
            newPA.setText(list.get(i).getPalabra_a());
            newPB.setText(list.get(i).getPalabra_b());
            newCheck.setChecked(list.get(i).isCheck());
        }
});
```

*Figura 55 - Botón para ver detalles de la página anterior*

Si al editar se quiere añadir una nueva página, se obtiene cada dato del EditText, ImageView, etc. correspondiente y se inserta en la Base de Datos. Por el contrario, si lo que se ha hecho ha sido sustituir algún campo ya completado, se realizará un update de la Base de Datos.

Si lo que queremos es borrar un cuento, se presionará el botón de la papelera, en la aplicación, y esta acción mandará llamar al método que elimina el registro de la Base de Datos, incluyendo sus páginas.

Como final, explicaremos el desarrollo de la parte en la que se ejecute (o corre) el cuento en sí. Decir, ante todo, que el sistema que se ha implementado es muy parecido al usado anteriormente, pero con algunas diferencias

En las figuras 56 y 57, se muestra el algoritmo para poder pasar a la página siguiente y a la página anterior:

```
//Botón para pasar a la página siguiente
siguiente.setOnClickListener((view)  -> {
        Log.e("i3",Integer.toString(i));
        c = db.rawQuery("Select * from pagina where pagina='"+list.get(i).getNumeroPag()+"'",null);
        if(c!=null){
            if(c.moveToFirst()){
                do{
                    numero_pagina = c.getInt(2);
                    Log.e("numero_pgina",Integer.toString(numero_pagina));
                    a1 = c.getInt(6);
                    b1 = c.getInt(7);
                }while (c.moveToNext());
            }
        }
        if(a1 == 0 && b1 == 0){
            Toast.makeText(context, "FIN DEL CUENTO", Toast.LENGTH_SHORT).show();
        }else{
            showMsg(context,a1,b1);
        }
        pagWIC = numero_pagina;
        Log.e("PagWIC",Integer.toString(numero_pagina));
});
```

*Figura 56 - Botón continuar la historia en página siguiente*

```
//Botón para volver a la página anterior
atras.setOnClickListener((view)  -> {
        int na = pagWIC;
        na--;
        int nb = pagWIC - 1;
        Bitmap bitmap = BitmapFactory.decodeByteArray(list.get(na).getFoto(),0,list.get(na).getFoto().length);
        imageCuentoC.setImageBitmap(bitmap);
        txtComent.setText(list.get(na).getComentario());
        i = list.get(nb).getNumeroPag();
        i--;
        Log.e("i2",Integer.toString(i));
});
```

*Figura 57 - Botón volver a la página anterior de la historia*

Como vemos en la figura 57, una vez que hemos obtenido el número de la página donde surgirían los dos caminos (largo y corto), a continuación, se llamará a un método que colocará la información de la página, tal y como indica la figura 58:

```
case 0:
    if(a == 0){
        Toast.makeText(context, "Fin del cuento", Toast.LENGTH_SHORT).show();
    }else{
        int n = a-1;
        Bitmap bitmap = BitmapFactory.decodeByteArray(list.get(n).getFoto(),0,list.get(n).getFoto().length);
        imageCuentoC.setImageBitmap(bitmap);
        txtComent.setText(list.get(n).getComentario());
        i = a-1;
    }
    dialogInterface.cancel();
    break;
case 1:
    if(b == 0){
        Toast.makeText(context, "Fin del cuento", Toast.LENGTH_SHORT).show();
    }else{
        int n1 = b-1;
        Bitmap bitmap1 = BitmapFactory.decodeByteArray(list.get(n1).getFoto(),0,list.get(n1).getFoto().length);
        imageCuentoC.setImageBitmap(bitmap1);
        txtComent.setText(list.get(n1).getComentario());
        i=b-1;
    }
    dialogInterface.cancel();
    break;
```

*Figura 58 - Poner páginas según tipo de camino*

Como ya hemos comentado, se trata de una aplicación interactiva en el que la aplicación actuará de narrador para contar la historia. En la figura 59 se muestra cómo se ha configurado el motor de síntesis de voz:

```
//Botón de TextToSpeech
textToSpeech = new TextToSpeech(context,this);
escucharRelato.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        textToSpeech.setLanguage(new Locale("spa", "ESP"));
        speak(list.get(i).getComentario());
    }
});
```

*Figura 59 - Botón para escuchar la historia*

Además de poder pasar páginas mediante el teclado, también es posible mediante el reconocimiento de voz, eligiendo uno u otro camino dependiendo de la respuesta que ofrezca el usuario. En las figuras 60, 61 y 62 se muestra la configuración del botón del micrófono, la del propio reconocedor de voz y una parte del código en el que se hace la comparación entre la palabra que dice el usuario y la guardada en la Base de Datos:

```
//Botón para el Micrófono
microphone.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        promptSpeechInput();
    }
});
```

*Figura 60 - Botón para el micrófono*

```
//Funcion para abrir el Recognizer
private void promptSpeechInput() {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
            "Say something...");
    try {
        startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
    } catch (ActivityNotFoundException a) {
        Toast.makeText(getApplicationContext(),
                "Sorry! Your device doesn't support speech input",
                Toast.LENGTH_SHORT).show();
    }
}
```

*Figura 61 - Reconocedor de voz*

```
if(result.get(0).equals(palabra1)){
    c = db.rawQuery("Select * from pagina where pagina='"+list.get(i).getNumeroPag()+"'",null);
    if(c!=null){
        if(c.moveToFirst()){
            do{
                numero_pagina = c.getInt(2);
                Log.e("numero_pgina",Integer.toString(numero_pagina));
                a1 = c.getInt(6);
                b1 = c.getInt(7);
            }while (c.moveToNext());
        }
    }
    if(a1 == 0 && b1 == 0){
        Toast.makeText(context, "FIN DEL CUENTO", Toast.LENGTH_SHORT).show();
    }else{
        if(a1 == 0){
            Toast.makeText(context, "Fin del cuento", Toast.LENGTH_SHORT).show();
        }else{
            int n = a1-1;
            Bitmap bitmap = BitmapFactory.decodeByteArray(list.get(n).getFoto(),0,list.get(n).getFoto().length);
            imageCuentoC.setImageBitmap(bitmap);
            txtComent.setText(list.get(n).getComentario());
            i = a1-1;
        }
    }
    pagWIC = numero_pagina;
    Log.e("PagWIC",Integer.toString(numero_pagina));
```

*Figura 62 - Comparación entre palabra reconocida y guardada*

# CAPÍTULO 4 – EVALUACIÓN

Este capítulo podemos dividirlo en tres partes: el proceso por el que se han realizado las pruebas tanto por parte del desarrollador como de los usuarios, la creación de una encuesta para conocer su opinión sobre la aplicación y las posibles y los resultados obtenidos tras su realización.

## 4.1. PRUEBAS

Este punto lo dividiremos en 2 partes:

- Pruebas del desarrollador

- Pruebas de los usuarios

### 4.1.1. PRUEBAS DEL DESARROLLADOR

Cuando hablamos de pruebas de desarrollador, nos referimos a todas las pruebas que se han tenido que realizar durante el proceso de programación de la aplicación, es decir:

- Pruebas con el emulador de Android Studio para comprobar la funcionalidad del trabajo.

- Pruebas en la generación del archivo .apk y su correspondiente instalación en los diferentes dispositivos.

- Pruebas de la aplicación en el propio terminal.

La creación de Cuentix ha sido larga, por lo que era necesario realizar un proceso de prueba constante con el fin de comprobar que las funcionalidades de la aplicación iban creándose y ejecutándose de forma óptima y como se esperaba.

No se puede hablar de pruebas finales de desarrollo hasta llegar a la última de todo el gran número de pruebas que se tuvieron que realizar, parte de ellas por no tener un nivel experto de conocimientos del lenguaje de programación y otra parte debido al volumen de líneas de código y la elevada complicidad de la aplicación.

### 4.1.2. PRUEBAS DE LOS USUARIOS

El orden que se llevó a cabo para el proceso de prueba por parte de los usuarios y que, posteriormente, pudieran realizar la encuesta mencionada, es el siguiente:

1. Instalación de la aplicación en el dispositivo móvil.
2. Selección del cuento “Caperucita Roja”.
3. Seleccionar la opción “Detalles”:
    - Observar los distintos puntos de la ventana.
    - Recorrido entre páginas.
    - Editar cuento y guardar los cambios.
4. Seleccionar la opción “Leer Cuento”:
    - Revisión de la estructura de la página.
    - Acción sobre botón “ESCUCHAR”.
    - Acción sobre botón “micrófono”.
    - Decir una de las dos respuestas que pregunta la aplicación.
    - Volver hacia la página anterior.
    - Paso a página siguiente mediante acción por teclado.
5. Crear un cuento nuevo:
    - Inserción de título y portada.
    - Guardar cuento.
    - Completar campos de página 1.
    - Guardar página 1 y continuar con las siguientes.
6. Operaciones de revisión de detalles y ejecución del cuento nuevo
    - Las mismas acciones que en los puntos 3 y 4.
7. Borrar el cuento creado en “Detalles”
8. Salida de la aplicación

Cabe destacar que este proceso se realizó para usuarios con una edad que superase los 15 años, dado que para el público infante se limitó a ejecutar el cuento y las acciones que se resumen en el punto 4.

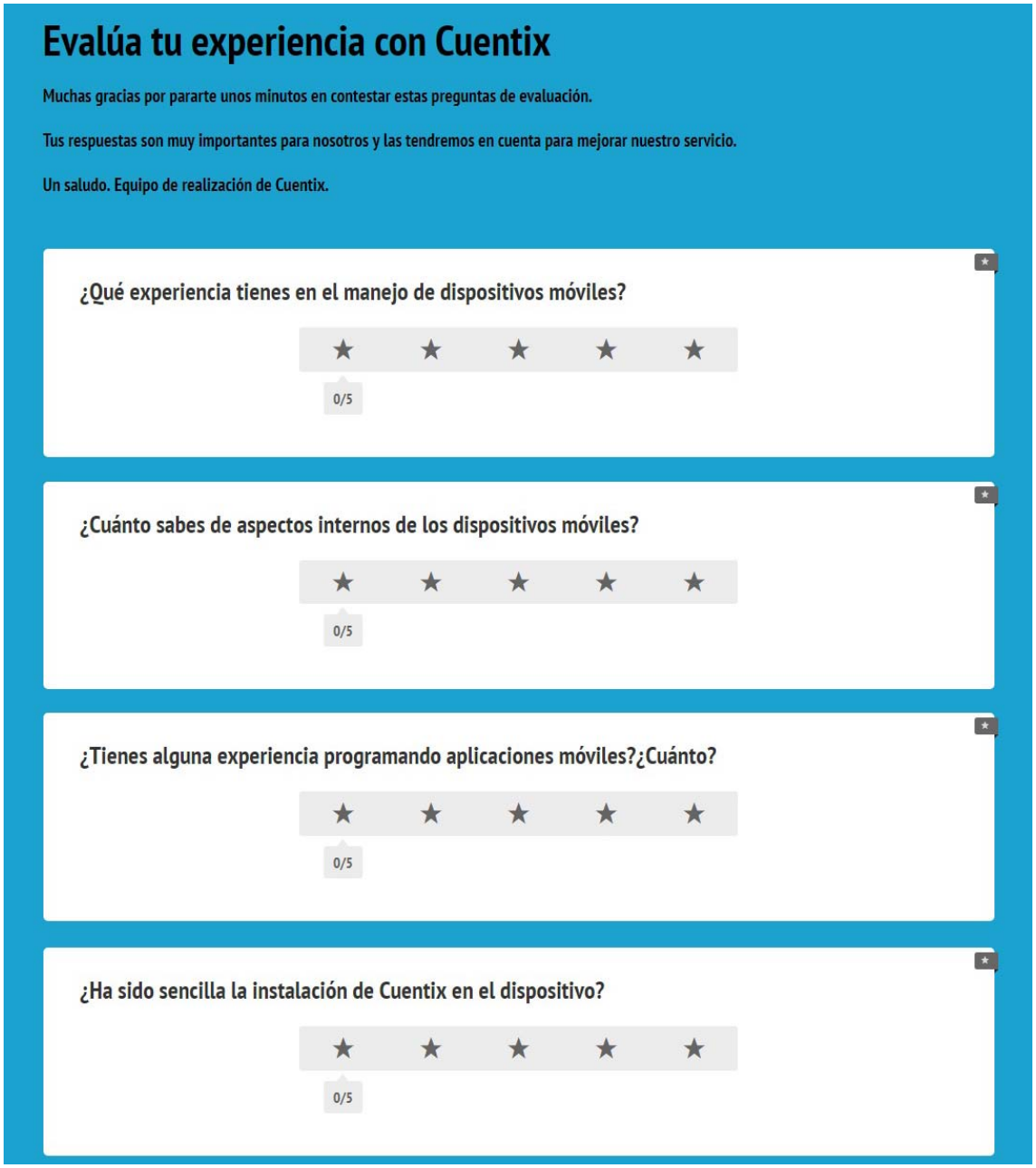
## 4.2. ENCUESTA A USUARIOS

La encuesta que se realizó a usuarios en edades comprendidas entre los 15 y los 90 años se creó mediante el servicio web Survio [35].

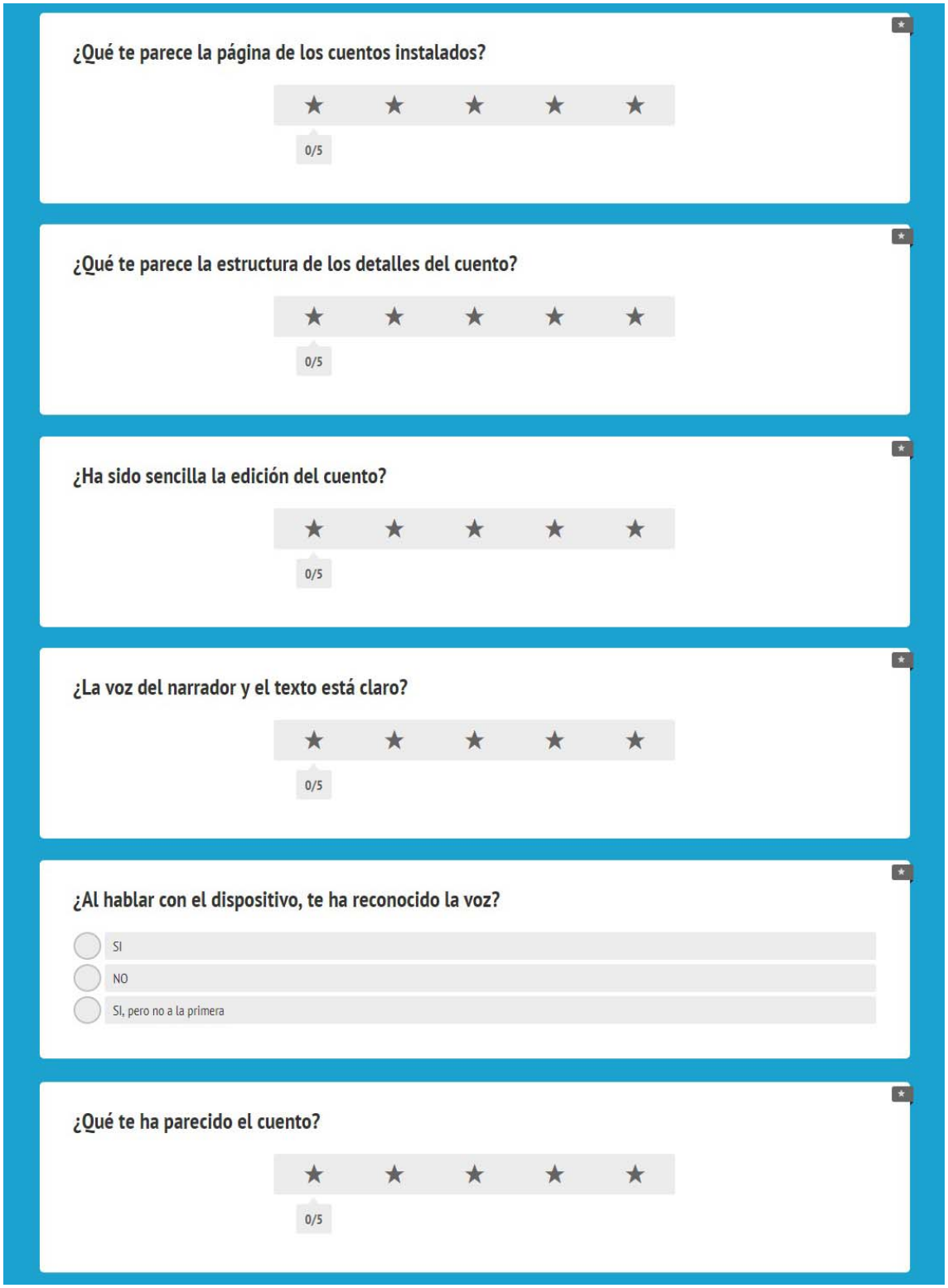
Todos los usuarios tuvieron que realizar las pruebas mencionadas en el punto 4.1.2 y, a continuación, realizar la encuesta proporcionada en el siguiente link:

https://www.survio.com/survey/d/A9G9B6O9X9K3O7G7M

Dicha encuesta y sus preguntas se ha diseño tan y como muestra la figura 63:



## 4. Evaluación



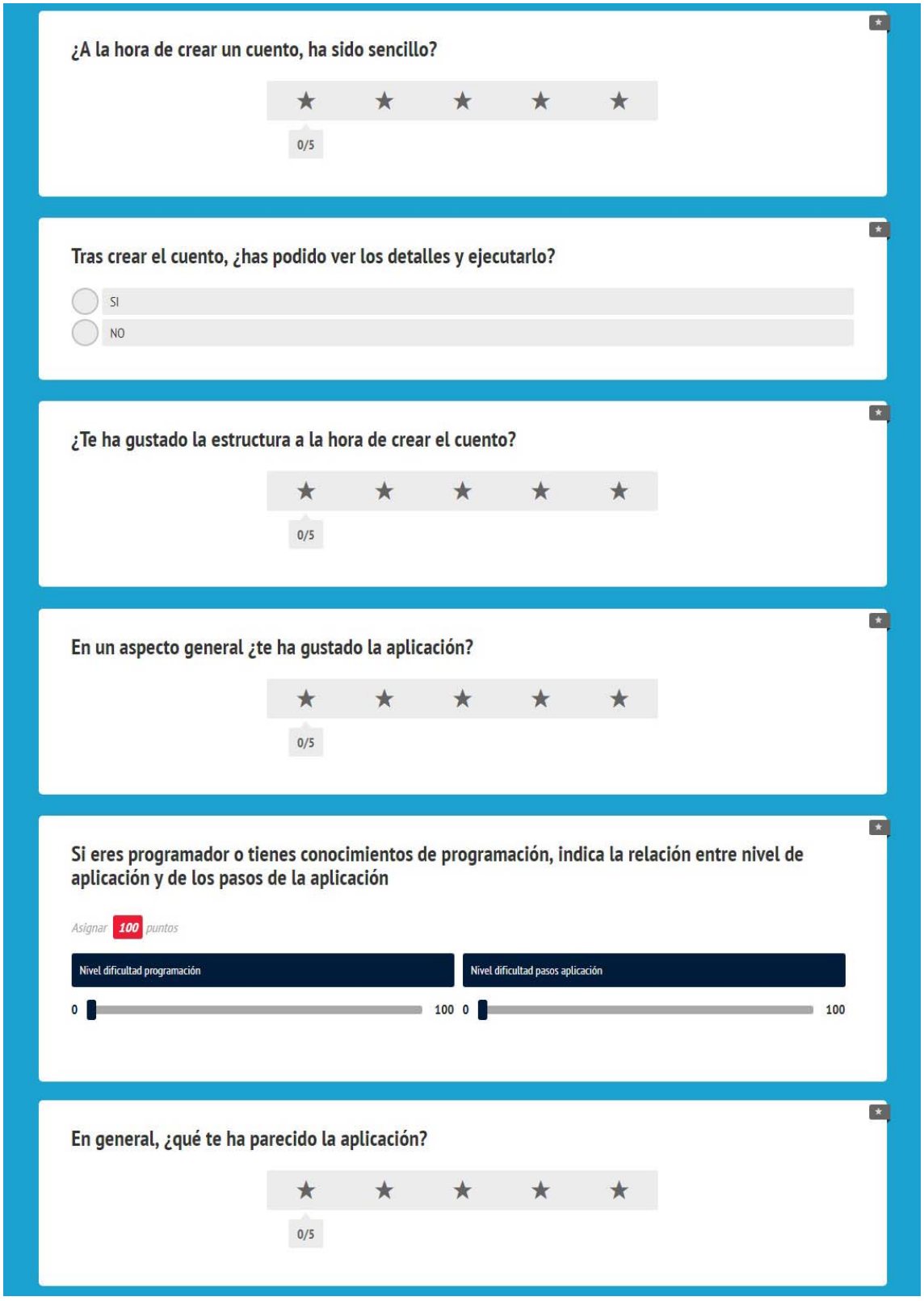

¿A la hora de crear un cuento, ha sido sencillo?
0/5
Tras crear el cuento, ¿has podido ver los detalles y ejecutarlo?
SI
NO
¿Te ha gustado la estructura a la hora de crear el cuento?
0/5
En un aspecto general ¿te ha gustado la aplicación?
0/5
Si eres programador o tienes conocimientos de programación, indica la relación entre nivel de aplicación y de los pasos de la aplicación
Asignar 100 puntos
Nivel dificultad programación
0
100
Nivel dificultad pasos aplicación
0
100
En general, ¿qué te ha parecido la aplicación?
0/5


*Figura 63 - Encuesta de Cuentix a usuarios*

Tras completar la encuesta y seleccionar el botón "ENVIAR ENCUESTA", los resultados se guardarán en la base de datos del servidor web para que el desarrollador pueda comprobar los resultados obtenidos.

## 4.3. RESULTADOS TRAS LA ENCUESTA A USUARIOS

Tras la realización del proceso de pruebas de 12 usuarios con edades comprendidas entre los 20 y 85 años, completaron la encuesta mostrada en la figura 75.

Los resultados obtenidos son muy diversos; sin embargo, un porcentaje elevado coinciden en tres puntos:

- Alguna dificultad en la edición del cuento: Expresan que la forma de editar el cuento no siempre es entendible por el usuario y debería mejorar.

- Problemas con el motor de síntesis de voz: La voz del narrador es muy robótica.

- Imágenes de las páginas del cuento: Sería bueno si las imágenes son dinámicas.

Gracias a los resultados obtenidos durante la encuesta, que se proporcionarán en los dos siguientes capítulos, y a los comentarios de cada usuario, se podrán realizar mejoras en la aplicación para solventar estos problemas.

De todas las estadísticas recogidas mediante la encuesta se proporcionarán, dentro del modelo de 5 estrellas, los datos sobre los tres problemas sugeridos, la opinión sobre la aplicación y unas gráficas sobre estos datos. Todo esto se recoge en las figuras 64, 65 y 66:


*Figura 64 - Estadísticas de partes problemáticas*

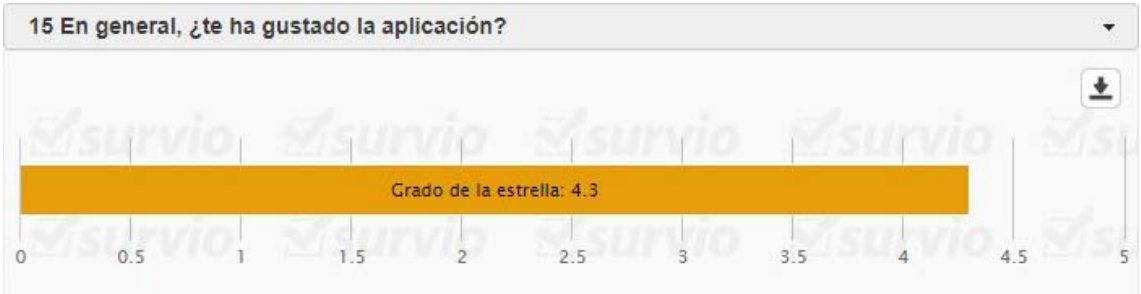



*Figura 65 - Estadística sobre la aplicación en general*

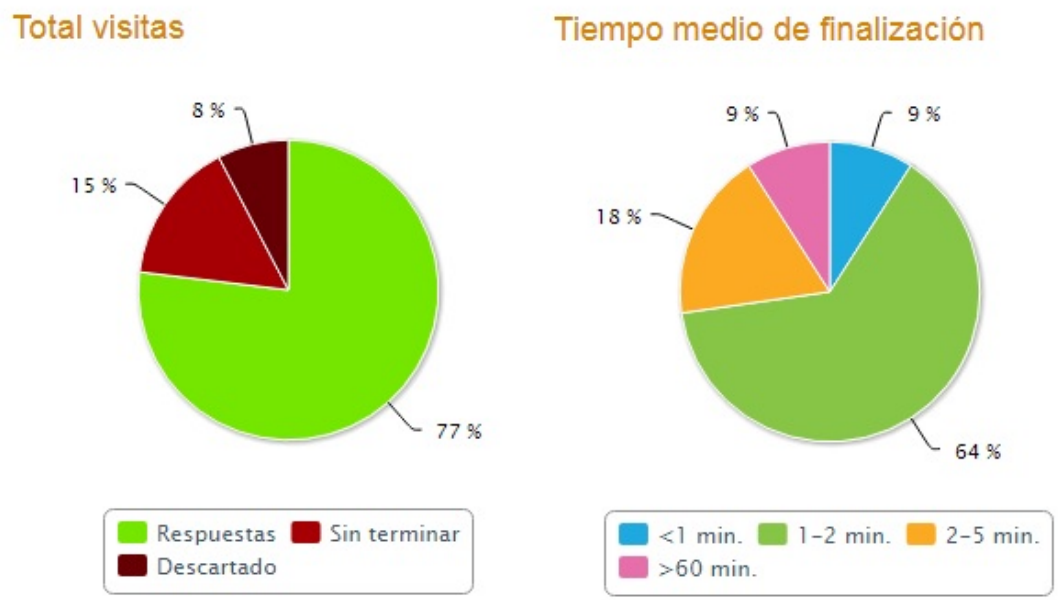



*Figura 66 - Estadísticas de la Encuesta*

# CAPÍTULO 5 – CONCLUSIONES Y TRABAJO FUTURO

Una vez finalizado el proyecto, memoria incluida, es hora de echar la vista atrás y pensar en el camino que se he recorrido.

Mucho tiempo ha pasado desde el primer momento en el que me senté en la silla de mi habitación pensando cómo iba a poder hacer la aplicación. Con unos conocimientos limitados y mucho esfuerzo y paciencia se ha conseguido no sólo realizar el trabajo, sino también adquirir nuevos conocimientos y nuevas formas de aprendizaje.

Comparando la idea inicia principal que tenía al comienzo con la que ahora tengo, podemos resumir las conclusiones a las que he llegado en los dos siguientes puntos.

## 5.1. CONCLUSIONES EN CUANTO A LA APLICACIÓN

Se proporcionarán las conclusiones, a modo de consejo, referentes a la realización de la aplicación:

- Nunca desesperes buscando una solución ante un problema, inténtalo una y otra vez hasta que lo consigas. Si no pudieras en ese momento, pasa a otro punto, pero nunca pares.

- Las mismas dudas que tienes tú, las ha tenido otra persona alguna vez, sólo necesitas encontrar la información en el lugar correcto.

- Lleva un orden lógico de los pasos a seguir para la realización de una función. Apúntatelo en un papel si es necesario, haz bocetos que te puedan ayudar.

- No sobrecargues una pantalla, limítate a que la aplicación haga lo que tiene que hacer.

- Tu aplicación no le gustará a todo el mundo, ten en cuenta su opinión para mejorar tu trabajo.

## 5.2. CONCLUSIONES EN CUANTO A LA MEMORIA

En este punto describiré las conclusiones a las que he llegado tras la realización de la presente memoria.

- Nunca estimes el tiempo que te va a llevar realizar algún capítulo, siempre será necesario dedicar el doble del tiempo

- Adopta una mentalidad relajada.

- Si te atascas, para 5 min, date una vuelta por casa y vuélvete a poner. Es necesario que tu cerebro respire un instante.

- Si no encuentras una palabra que decir, ve al diccionario y encuéntrala.

- Nunca pienses en lo que te queda, sino en lo que llevas hecho.

## 5.3. TRABAJO FUTURO

Todo trabajo se puede mejorar, sobre todo si se trata de cualquier proyecto tecnológico. Tras el inicio, realización y finalización de la aplicación Android he adquirido conocimientos no sólo del propio lenguaje de programación, sino también una visión más amplia sobre las aplicaciones que hay ahora mismo en el mercado. Gracias a estos conocimientos y a la ayuda de los usuarios que probaron y completaron la encuesta descrita en el capítulo 4, las siguientes líneas harán referencia a los posibles consejos, ideas y actualizaciones que podrían llevarse a cabo para mejorar la aplicación Cuentix:

**- Voz del narrador:** Dado que, para muchos usuarios, la voz que trae consigo el dispositivo (recordemos que es Google TTS) era demasiado robótica, podríamos solventar este problema mediante dos opciones. Una de ella podría ser la implementación de un nuevo motor de síntesis mucho más natural, que sea capaz de imitar la voz humana lo más cercanamente posible. La otra opción, más sencilla, es limitar la voz del narrador a clips de audio; sin embargo, esta acción suprimiría la acción del TTS ya que limitaríamos el cuento a clips de audio sin relación con el texto propio de la página del cuento.

**- Cuentos más dinámicos:** Para solucionarlo, se podrían usar varias imágenes en una misma página y dividiéndola en diferentes partes para conseguir más dinamismo en la propia página. Sería un punto bastante interesante de implementar ya que aumentaría el nivel de dificultad de la programación tanto como la atención del usuario.

**- Ventanas emergentes para editar y crear cuentos:** Dado que este problema es el más demandado por los usuarios, también es el más importante. Éstos tuvieron algunos problemas a la hora de encontrar el orden lógico de editar y crear, que se solucionarían rápidamente creando ventanas emergentes que aparecieran según el orden en que se debieran seguir las distintas acciones y desapareciendo cuando se completara cada una de ellas, dando paso a la siguiente ventana. Esta acción aumentaría la carga en la pantalla, pero disminuiría el grado de dificultad.

**- Aplicaciones sobre Cuentix:** La aplicación sólo se limita a crear, editar y leer cuentos pero, ¿y si fuera posible que Cuentix contuviese algo más? Tenemos la oportunidad de que la aplicación conlleve dentro de sí más aplicaciones: cuentacuentos, aprende a contar, aprende los animales, los colores. El límite lo pondríamos nosotros mismos.

**- Base de Datos y Servidor externos:** El proyecto se instala dentro de nuestra aplicación con un cuento por defecto, esto es común para todo usuario que descargue e instale la aplicación. Sin embargo, podemos crear una base de datos externa unida a un servidor web en el que el usuario, mediante una cuenta privada, pueda crear sus propios cuentos y subirlos a la nube, pueda decidir si quiere que otros usuarios puedan descargarse su cuento e incluso descargarse el cuento de otro usuario.

**- Integración completa del reconocimiento de voz:** Dado que se trata de un cuento interactivo, podemos externalizar a todo el cuento la idea del reconocimiento de voz sólo presente en la elección de caminos del cuento. Con esto, podríamos seleccionar el cuento que queremos leer, ver los detalles de una página y cambiarlos a nuestro antojo e incluso crear nuevos cuentos sin necesidad de tocar la pantalla.

## 5.4. OPINION PERSONAL

Mucha gente, yo me incluía antes de empezarlo, piensa que el TFG es un mero formalismo, la asignatura “fácil” que recompensa todos los años y horas que he tenido que dedicar a la realización de la carrera; pero no es así. Este trabajo es muy difícil, sobre todo si tienes que compaginarlo con otras obligaciones, en el que a la mínima que lo dejes apartado, luego volver a él te cuesta dos o incluso tres veces más que al principio.

La elección del proyecto es una tarea que bien me recordó al momento de terminar selectividad y comenzar a buscar carreras que se adaptaran a mí. Miras con ilusión y esmero cuáles son los trabajos que hay presentados en el tablón, sus características, para que fin se van a realizar y las herramientas necesarias. Siempre se intenta buscar un proyecto que sea relativamente sencillo y que no se tarde mucho en poder realizar porque, no nos engañemos, hay más vida a parte de la universidad. En mi caso he tenido que lidiar con el proyecto, algunas asignaturas que se quedaron conmigo más tiempo del que me gustaría decir, 10 horas diarias de trabajo (o de hasta 16 los sábados), participar en un equipo de fútbol. Todos estos “problemas” (son problemas más que soluciones), no hacían más que llenar la mochila que luego tendría que llevar para subir la montaña de mi objetivo, terminar el proyecto.

Como he dicho anteriormente, la primera idea que se tiene al comenzar a mirar el TFG es elegir un proyecto que sea rápido y sencillo, ya que tras 4,5 o 6 años de carrera lo que se quiere es terminar y cerrar un ciclo ya sea para estar tranquilo con el título “debajo del brazo” o para abrirte puertas a otros objetivos. Esta idea, una vez elegido el proyecto que queremos hacer, va tornando en otra muy distinta, en que si las dos últimas siglas de TFG significan Fin de Grado, es porque no es un mero trámite.

Pero no todo son penas. La búsqueda del TFG fue más orientado hacia la programación desde un principio ya que, tras pasar por las 39 asignaturas que componen la carrera de Sistemas de Comunicaciones, las asignaturas que más me gustaron fueron las de programación. Tengo que decir que la idea de, mediante palabras, llegar a construir algo mucho más grande y útil para las personas me produce una sensación especial.

Gracias a este proyecto, he tenido la oportunidad de poder volver a una rutina de aprendizaje que poco a poco estaba perdiendo y, sobretodo, de poder introducirme en el mundo de la programación en Android que tanto he querido aprender.

Como he mencionado en las líneas anteriores, hay dos posibles opciones a la hora de concluir la carrera: termino la carrera y se acabó, o sigo estudiando y aprendiendo gracias a los conocimientos obtenidos. En mi caso, siendo una persona a la que le gusta estar en constante aprendizaje, diré que todos estos años en la carrera, unida a la experiencia vivida con el TFG, han hecho que me decida a seguir estudiando, a reforzar mis conocimientos sobre Android hasta el punto de poder realizar cualquier aplicación de forma sencilla.

Por último, y a modo de resumen, diré que la realización del proyecto ha sido dura, con bastantes momentos de altibajos, con muchas conversaciones familiares, muchos apoyos pero también muchas discusiones, pero también de mucho aprendizaje, reafirmación de ideas y también de cambios. Quiero volver a dar las gracias a toda mi familia, mi pareja, amigos que siguen y los que se fueron y profesores por haberme dado la oportunidad de crecer como persona, haberme dado su apoyo y jamás haberme dado por perdido

# CAPÍTULO 6 – GESTIÓN DEL PROYECTO

## 6.1. MARCO REGULADOR

El éxito de las aplicaciones móviles se debe a que están al alcance de casi todos los bolsillos, ya que gran parte de ellas son gratuitas, y al alcance de todo tipo de personas. Además su éxito radica en que no sólo se dirigen a smartphones, sino cada vez hay más aplicaciones dirigidas a otros dispositivos como Tablets, Smart TVs, etc.

La instalación de aplicaciones puede recoger gran cantidad de datos personales; por este motivo, las autoridades europeas de Protección de Datos ratificaron el primer informe que aclara el marco jurídico aplicable al uso de Apps para dispositivos inteligentes, concluyendo que será aplicable a los desarrolladores de las aplicaciones. No cumplir con estos requisitos básicos puede generar problemas con la administración, la justicia y los usuarios.

La siguiente lista de requisitos legales fue elaborada por el bufete Ad&Law [36]:

- **Permisos, licencia y condiciones de uso:** Hay que ser claros y explícitos a la hora de solicitar permisos al usuario. Además, es obligatorio desarrollar licencias y condiciones de uso.

- **Derechos propios y de terceros:** Es obligatorio disponer de licencias de los recursos que se vayan a utilizar, leyendo detenidamente las condiciones. Además, conviene proteger el contenido para evitar plagios y copias.

- **Privacidad:** La recogida de información del usuario debe ser la indispensable para el funcionamiento de la App y éste debe tener la posibilidad de configurar la privacidad.

- **Información y cookies:** Es fundamental informar al usuario de los aspectos regulados en la ley y mostrar los datos sobre los creadores y sobre quienes se encuentra tras la App. También es necesario que el usuario acepte las cookies, mediante un aviso informativo con la información básica y precisa sobre las mismas, y los aspectos exigidos por la ley.

- **Markets:** Tienen condiciones muy estrictas para que se puedan publicar las aplicaciones por lo que hay que cumplir siempre lo que piden. De hecho, incluso cumpliendo las condiciones al colgar la app, éstas pueden cambiar y hacer que la aplicación no esté disponible para usuarios nuevos.

- **Publicidad:** Si monetizamos una aplicación a través de publicidad, ésta debe identificarse siempre como tal.

Conociendo estos requisitos, podremos redactar los términos legales necesarios para nuestra aplicación:

- Dado que la aplicación abrirá tanto la galería del dispositivo como la de la tarjeta externa que se tenga instalada, será necesario que el usuario acepte el término de “Acceso al dispositivo” antes de instalar la app.

- No será necesario ningún término adicional de privacidad dado que la aplicación no usa Bases de Datos externas ni usa enlaces a páginas web.

- Suponiendo que la aplicación se haya subido a “Google Play”, será necesario cumplir los requisitos obligatorios por parte de la empresa Google [37].

En la figura 67 se muestran los requisitos legales que es necesario que el usuario acepte para poder usar la aplicación:

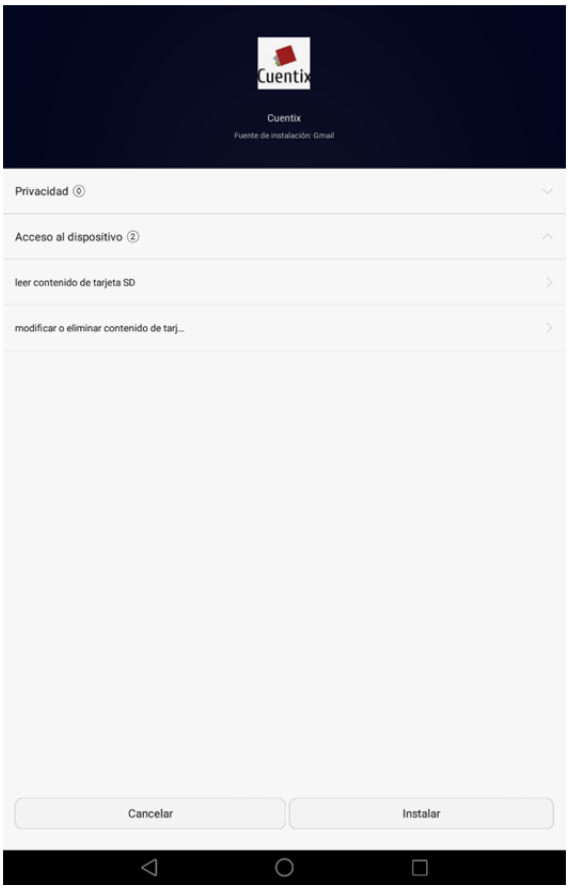



*Figura 67 - Requisitos legales Cuentix*

## 6.2. COSTES

En este capítulo se realizara el estudio del presupuesto necesario para la realización del proyecto en su totalidad, teniendo en cuenta recursos, horas de trabajo, costes de personal, etc.

Los costes son todos aquellos gastos en los que incurre una empresa para realizar una tarea, un trabajo o un proyecto determinado [38]. Se pueden dividir en dos partes:

- Costes directos

- Costes indirectos

### 6.2.1. COSTES DIRECTOS

Son los que guardan una relación estrecha con el producto o servicio. Se establecen desde las primeras fases de producción y suelen reflejarse en los presupuestos o estimaciones de costos. También los que se relacionan con la mano de obra directa, por ejemplo, el pago que reciben las personas que trabajan en el proyecto.

Atendiendo a esta definición y orientándola a nuestro proyecto, podemos seccionar los costes directos en:

- Coste de personal

- Coste de recursos

#### 6.2.1.1. Costes de personal

Estos serían los costes referidos al empleado, tanto las horas trabajadas como el coste debido a su puesto de trabajo.

- Horas trabajadas: Las horas dedicadas al proyecto. En este caso se ha tardado 50 días, 5 horas al día.

- Costes de gremio: Estos costes variarán según el puesto del empleado y del contrato con la propia empresa. Suponiendo un puesto de ingeniero estándar, podemos estimar un sueldo de 21000€ al año.

Para realizar el cálculo de los costes de personal, será necesario conocer cuánto le cuesta el empleado a la propia empresa. Según la normativa vigente y, suponiendo que se trata de un empleado estándar, el coste al año para la empresa será de 30000€ al año [39].

Ahora podemos calcular las horas totales trabajadas en el proyecto y el precio por hora del empleado con las fórmulas 1 y 2:

- Horas trabajadas: 70 días x 5 horas/día = 350 horas

- Coste de gremio: 30000 €/año / 2080 horas/año = 14,43 €/hora

$$Horas\ totales\ trabajadas\ (horas) = días\ trabajados\ x\ \frac{horas\ trabajadas}{día}$$

*Fórmula 1 - Horas trabajadas en el proyecto*

$$Costes\ de\ gremio\ \left(\frac{€}{hora}\right) = \frac{sueldo\ (€)}{año} \div \frac{horas\ laborables}{año}$$

*Fórmula 2 - Costes de gremio*

Una vez que ya tenemos las horas trabajadas y los costes de gremio, podremos calcular los costes de personal, mediante la fórmula 3:

- Costes de personal: 350 horas x 14,43 €/hora = 5050,5 €

$$Costes\ de\ personal\ (€) = horas\ trabajadas\ \times costes\ de\ gremio$$

*Fórmula 3 - Costes de personal*

Como vemos, los costes de personal ascienden a **5050,5 €**

### 6.2.1.2. Costes de recursos

Estos costes son debido al material, tanto hardware como software, utilizado para la realización de la aplicación. Como comprobaremos, no hemos incluido el coste del Software debido a que hemos supuesto que el coste es cero.

La siguiente tabla refleja estos costes:

| RECURSO | COSTE (€) |
|---|---|
| **Ordenador HP Intel Core 2 DUO** | 500 |
| **Huawei Mediapad M2 10** | 250 |
| **Samsung Galaxy A5** | 200 |
| **Monitor HP** | 100 |

*Tabla 15- Costes de cada recurso utilizado*

Según la ley, no se puede imputar el gasto completo de cada recurso utilizado, ya que la vida útil no coincide con la duración del proyecto. Normalmente la vida útil que se tiene en cuenta es de 4 años, por lo que la tabla 15 se actualiza a la tabla 16, expuesta a continuación:

| RECURSO | TIEMPO DEDICADO (DÍA) | VIDA ÚTIL (DÍA) | COSTE (€) | COSTE IMPUTABLE (€) |
|---|---|---|---|---|
| **Ordenador HP Intel Core 2 DUO** | 70 | 1461 | 500 | 24 |
| **Huawei Mediapad M2 10** | 70 | 1461 | 250 | 12 |
| **Samsung Galaxy A5** | 70 | 1461 | 200 | 9,6 |
| **Monitor HP** | 70 | 1461 | 100 | 4,8 |

*Tabla 16 - Costes de recursos imputables*

Para calcular el coste imputable, se ha usado la siguiente fórmula:

$$Costes\ imputables\ (€) = \frac{Tiempo\ dedicado\ (día)}{Vida\ útil\ (día)} \times Costes\ (€)$$

*Fórmula 4 - Costes de recursos imputables*

Sumando estos costes, el resultado total de costes de recursos será de 50,4 €

### 6.2.2. COSTES INDIRECTOS

Estos costes son los que se relacionan tangencialmente con los proyectos o las tareas previstas. También debemos incluir los costes indirectos generales del tipo administrativo o financiero.

$$Costes\ indirectos = Costes\ directos \times 0{,}25$$

*Fórmula 5 - Costes indirectos*

En nuestro caso, los costes indirectos serán un 25% de los costes directos. Por lo tanto, siguiendo la fórmula 4, los costes se reflejan en la siguiente tabla:

| TIPO DE COSTE | COSTE DIRECTO (€) | COSTE INDIRECTO (€) |
|---|---|---|
| **Costes de personal** | 5050,5 | 1262,63 |
| **Costes de recursos** | 50,4 | 12,6 |

*Tabla 17 - Costes indirectos totales*

Por lo tanto, los costes indirectos ascienden a **1275,23 €**

### 6.2.3. COSTE FINAL

Una vez calculados los costes, es hora de calcular el coste final del proyecto. Para ellos, resumiremos todos los costes en la tabla 18, indicando el coste sin y con IVA:

| TIPO DE COSTE | COSTE (€) |
|---|---|
| **Costes directos** | 5100,9 |
| **Costes indirectos** | 1275,23 |
| **Coste total CON IVA** | 7715,12 |
| **Coste total SIN IVA** | 6376,13 |

*Tabla 18 - Coste Final del Proyecto*

Tras todos los datos presentados, podremos concluir que el coste final de realización del proyecto, con IVA, será de **7715,12 E**

## 6.3. IMPACTO SOCIO-ECONÓMICO

En esta sección se explicará el impacto tanto social como económico que provocaría la aplicación tras su puesta en servicio.

### 6.3.1. IMPACTO SOCIAL

Antes de nada, la primera consideración que hay que tener en cuenta es el tipo de aplicación que se está hablando: se trata de una aplicación de cuentacuentos para el público infantil. Además, según el modelo de las 4 etapas del desarrollo cognitivo infantil de Piaget, una de más importantes es la Etapa sensotiomotora, en la que el niño utiliza sus sentidos y capacidades motoras para conocer los objetos y el mundo. Esto quiere decir, con un tono más vulgar, que "todo lo que se mueva o suene, el niño le hace caso". A todo esto hay que añadirle que una de las finalidades de Cuentix, aparte de entretener, es la de fomentar la capacidad cognitiva y educacional del usuario; con la aplicación, conseguiremos que el niño se concentre más, mejorará su capacidad de razonamiento, perfeccionará su forma de hablar y de escuchar.

Teniendo esto en cuenta, debemos observar cuántas aplicaciones infantiles hay en el mercado. Este número es tremendamente elevado debido a que los padres intentan que sus hijos sean lo más felices posibles y, sobre todo, que estén distraídos.

Pero no sólo la aplicación está enfocada hacia los niños y sus padres, sino también a centros educacionales e, incluso, para hospitales. Con Cuentix, se podrá tratar a pacientes con Alzheimer o enfermedades de degeneración cognitiva y cerebral, mediante la creación de sus propios cuentos, recordando cuentos ya conocidos por ellos, contar sus propias historias.

Si juntamos todos estos aspectos, podemos decir que el impacto que tenga la aplicación en la sociedad será muy positiva, ya que se podrá usar para muy variadas situaciones, ayudando a todo tipo de persona y, en especial, a pasar un buen rato.

### 6.3.2. IMPACTO ECONÓMICO

Primeramente lo que debo decir ante este punto es que la creación y el desarrollo de Cuentix se realizaron con la firme idea de que su uso, por parte de los usuarios, siempre fuese gratuito.

Se trata de una aplicación en un lenguaje de programación gratuito, para dispositivos móviles con un Sistema Operativo de coste cero y destinada a público de cualquier edad y condición social, independientemente de su poder adquisitivo. Por estos motivos, cuando hablamos de impacto económico, no podremos decir más que será el producido por el coste de realización y de material expuesto en el capítulo 7.

El desarrollo de Cuentix, dado que está creado en Android que es un lenguaje de código libre, puede generar el desarrollo de otras aplicaciones por parte de desarrolladores que sí prefieran ponerle un precio a su trabajo. Esta sería la única relación que tendría la aplicación con cualquier idea económica en contra del usuario.

En el caso de que la aplicación, tras conseguir una popularidad tan elevada que llevase a una empresa a tomar la decisión de querer comprarla, ésta sería la responsable de ponerle un precio de cara a la venta al público pero siempre con un nombre distinto al de “Cuentix” dado que, como se ha mencionado anteriormente, su finalidad inicial no era la de coste alguno al usuario.

# SUMMARY IN ENGLISH

In this final chapter the most important sections have been translated, to provide an accurate description of the aim of my project.

## ABSTRACT

Since the dawn of humankind, we have left a mark on history, remembering past events by virtue of teaching customs and experiences. As our skills and technology have evolved, the way of outsourcing information has also adapted and improved. Nowadays in the 21st century, the so-called digital age, it is becoming easier and faster to transmit and share information among individuals.

Over the past years, the world has undergone a high and rapid technological development, succeeding in the integration itself in the areas of knowledge and production and in the mentality of the population. That has made it so essential that nobody would to be able to imagine the world without it.

In addition, with the arrival of ¡ computers, a boom period began for the new technologies. Due to the great social impact it had on society, the pursuit of excellence led to an improvement of the products and the proximity of them to all social classes.

In this project I have developed an application of Android storytellers through Android Studio. As this application is intended for children, the main objective of the application is to offer a straightforward and joyful interface.

The story consists in 4 parts:

- Story list: All stories that have been saved in the device are displayed.

- Create a new story: The user can create his own story using his own pictures and writing the comments.

- Edit a saved story: We can change the story we want and when we want.

- Read a story: The application will read the story selected.

# OBJECTIVES

Since this project is only an update of the famous books "tell your story", an important objective was to make it simple, yet interesting for the target group, i.e. the children.

However, to achieve this goal, clear and extensive knowledge is required about Android programming, which was acquired through extensive reading of the existing literature.

At this point, the most important aspect is how the application should be done: do we want it to be dynamic, speak to the user, that we could talk to the application or could we only interact with a touch-user interface?

After deliberating the proposed options, we were able to decide how the application would be, namely:

- It should be an application that does not bore users.

- It should be an application that could be used both by voice and touch-user interface (children have a preference for interactives devices that allow touch).

- It should be be simple in design, so children can use without difficulties, and yet allow to read a story and to create a new one.

Once the problem was solved, it was only to integrate all the points, and write the correct commands, following a double-checking process to fix the errors that were found. As a result, "***Cuentix***" was created.

In short, the main objective of this project is not only to "digitize" a story, but also to make way for other new applications and updates that can improve what has already been created and even give ideas for other possible more ambitious projects.

# EVALUATION

There are two testing options: developer testing and user testing.

The developer tests are all tests that had to be performed during the application programming process).

The user tests follow this order:

1. Installation of the application on the mobile device.
2. Selection of the story "Little Red Riding Hood".
3. Selection and revision of "Details".
4. Select the "Read Story".
5. Create a new story.
6. Selection "Details" and "Read Story" of the new story.
7. Delete the new story.
8. Exit the application.

To end the tests, users aged between 15 and 90 were asked to answer a questionnaire in order to know their opinion, errors and possible improvements of ***Cuentix***.

The results obtained are very diverse; however, a high percentage coincide in three points:

- Some difficulty in editing the story: They express that the way of editing the story is not always understandable by the user and should improve.

- Problems with the engine of synthesis of voice: The voice of the narrator is very robotic.

- Images of the pages of the story: It would be good if the images were dynamic.

In addition, the application was liked by the majority of the users who performed the tests.

# CONCLUSIONS

A lot of time has passed since this project was conceived and extensive time was employed in brainstorming to bringing it to the current state. With effort and patience, I was able to success and also on the way I acquired valuable knowledge and new forms of learning.

Comparing the main idea I had at the beginning with the one I have now, we can summarize the conclusions that I have reached in the following points.

As for the application:

- Never despair looking for a solution to a problem, try again and again until you get it. If you could not at that time, move on to another point, but never stop.

- The same doubts you have, you have had someone else, you just need to find the information in the right place.

- It takes a logical order of the steps to follow for the accomplishment of a function. Put it on paper if necessary, make sketches that can help you.

- Do not overload a screen, just let the application do what it has to do.

- Your application will not please everyone, consider your opinion to improve your work.

As for the memory:

- Never estimate how long it will take you to do a chapter, it will always be necessary to devote twice as much time

- Adopt a relaxed mentality.

- If you get stuck, for 5 min, go around the house and put it back. Your brain needs to breathe for a moment. Long-term productivity rates higher above short-term productivity.

- If you cannot find a word to say, go to the dictionary and find it.

- Never think about what you have left, but what you have done.

# FUTURE WORK

All work can be improved, especially any technological project. After the start, completion of the Android application I have acquired knowledge not only of the programming language itself, but also a broader view of the applications that are now on the market. Thanks to this knowledge and to the help of the users who tested and completed the survey described in Chapter 4, the following comments will refer to the possible tips, ideas and updates that could be made to improve the ***Cuentix*** application:

**- Narrator voice:** Because for many users, the voice that comes with the device (i.e. Google TTS) was too robotic, we could solve this problem with two different approaches. One of them could be the implementation of a new, much more natural synthesis engine that is able to imitate the human voice as closely as possible. The other, simpler option is to limit the voice of the narrator to audio clips; however, this action would suppress the action of the TTS since we would limit the story to audio clips unrelated to the text itself of the page of the story.

**- More dynamic stories:** To solve this, we could use several images on one page and divide it into different parts to get more dynamism on the page itself. It would be a very interesting point to implement since it would increase the level of difficulty of programming as much as the attention of the user.

**- Pop-up windows to edit and create stories:** Since this problem is the most demanded by users, it is also the most important. These had some problems in finding the logical order to edit and create, which would be solved quickly creating pop-ups that appeared in the order in which the different actions should be followed and disappearing when each one was completed, giving way to the next window. This action would increase the load on the screen, but would decrease the degree of difficulty.

**- Applications contained in Cuentix:** The application is limited to creating, editing and reading stories, but what if it were possible that ***Cuentix*** contained something else? We have the opportunity for the application to carry within itself more applications: storytelling, learning to count, learning animals and colors. We would put the limit ourselves.

**- External Database and Server:** The project is installed inside our application with a default story, this is common for all users who download and install the application. However, we can create an external database linked to a web server in which the user, through a private account, can create his own stories and upload them to the cloud, can decide if he wants other users to download his story and even download the story of another user.

**- Full integration of voice recognition:** Since this is an interactive story, we can outsource the whole story to the idea of voice recognition only present in the choice of ways of the story. With this, we could select the story we want to read, view the details of a page and change them at our whim and even create new stories without touching the screen.

## PERSONAL OPINION

Many people think that the TFG is a formalism, the "easy" subject that rewards all the years and hours that I have had to dedicate to the realization of the race; but it's not like that. This work is very difficult, especially if you have to combine it with other obligations, in which to the minimum that you leave it away, then return to it costs you two or even three times more than at the beginning.

The choice of the project is a task that well registered me at the time of finishing SAT and start looking for careers that fit me. You look with enthusiasm and dedication what are the works that are present in the board, its characteristics, so that an accomplishment and the necessary tools. We always try to look for a project that the sea is relatively simple and it does not take long to be able to do it because, let's not fool ourselves, there is more life apart from the university. In the event that I had to deal with the project, some subjects that stayed with me longer than I would like to say, 10 hours a day of work (or 16 Saturdays), participate in a football team. All these "problems", there is nothing more to fill the backpack that would later have to take to climb the mountain of my goal, finish the project.

As I said before, the first idea you have when you start looking at the TFG is to choose a project that is fast and easy, since after 4, 5 or 6 years of career what you want is to finish and close a cycle either to be quiet under the title "under the arm" or to open doors to other objectives. This idea, once chosen the project we want to do, is becoming a very different one, in that if the last two acronyms of TFG mean End of Grade, it is because it is not a mere process.

But not everything is pain. The search for the TFG was more oriented to programming from the beginning since, after going through the 39 subjects that compose the career of Communications Systems, the subjects that I liked the most were the programming. I have to say that the idea of using words to build something much bigger and useful for people gives me a special feeling.

Thanks to this project, I have had the opportunity to return to a learning routine that I was slowly losing and, above all, to be able to introduce myself to the world of programming in Android that I have wanted so much to learn.

As I mentioned in the previous lines, there are two possible options at the time of finishing the race: I finished the race and finished, or I continue studying and learning thanks to the obtained knowledge. In my case, being a person who likes to be in constant learning, I will say that all these years in the race, coupled with the experience lived with the TFG, have made me decide to continue studying, to reinforce my knowledge on Android to the point of being able to make any application in a simple way.

# BIBLIOGRAFÍA

[1] Tomás, Jesús. “Seguridad y posicionamiento. El gran Libro de Android”. 5ª Edición. 110-130. 2016.

[2] “KitKat quiere traer a Android para todos”, disponible en: https://www.cnet.com/es/analisis/google-android-kitkat/ [Último acceso: septiembre 2017]

[3] “Características resaltantes del Android 4.4 KitKat”, disponible en: https://www.aboutespanol.com/caracteristicas-resultantes-del-android-4-4-kitkat-580827 [Último acceso: septiembre 2017]

[4] “Android Lollipop: una actualización con mucho más que Diseño Material”, disponible en: https://www.cnet.com/es/analisis/google-android-5-0-lollipop/ [Último acceso: septiembre 2017]

[5] “16 cosas que puedes hacer en Android Lollipop y no podías en KitKat”, disponible en: http://es.gizmodo.com/16-cosas-que-puedes-hacer-en-lollipop-y-no-podias-hacer-1659755788 [Último acceso: septiembre 2017]

[6] “Android 6.0 Marshmallow pule Android y mejora su desempeño”, disponible en: https://www.cnet.com/es/analisis/google-android-6-0-marshmallow/ [Último acceso: septiembre 2017]

[7] “Android 6.0 Marshmallow pule Android y mejora su desempeño”, disponible en: https://www.tuexperto.com/2015/12/01/7-cosas-que-puedes-hacer-en-android-6-0-que-no-podias-hacer-en-android-5-0/ [Último acceso: septiembre 2017]

[8] “Android Nougat: análisis”, disponible en: https://www.cnet.com/es/analisis/google-android-nougat/resena/ [Último acceso: septiembre 2017]

[9] “15 mejoras de Android 7.0 Nougat que las versiones anteriores no tenían”, disponible en: https://elandroidelibre.elespanol.com/2016/09/15-mejoras-nougat-marshmallow-no-tenia.html [Último acceso: septiembre 2017]

[10] “Qué es Project Treble y cómo afectará a las ROMs en el futuro”, disponible en: http://www.proandroid.com/project-treble-afectara-las-rom-futuro/ [Último acceso: septiembre 2017]

[11] “Android 8.0 Oreo ofrece más fluidez, actualizaciones más rápidas y otras novedades”, disponible en: https://www.cnet.com/es/analisis/android-o/primer-vistazo/ [Último acceso: septiembre 2017]

[12] “Android 8.0 ya es oficial: estas son sus 23 novedades más destacadas”, disponible en: https://www.xatakandroid.com/sistema-operativo/android-8-0-oreo-novedades-y-caracteristicas [Último acceso: septiembre 2017]

[13] “¿Para qué versión de Android debo desarrollar?”, disponible en: https://geekytheory.com/para-que-version-de-android-debo-desarrollar [Último acceso: septiembre 2017]

[14] “Conoce Android Studio”, disponible en: https://developer.android.com/studio/intro/index.html?hl=es-419 [Último acceso: septiembre 2017]

[15] “Android Studio v1.0: características y comparativa con Eclipse”, disponible en: https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/ [Último acceso: septiembre 2017]

[16] Sierra, Alberto. “Aprendiendo APP Inventor. Creando Apps de Android sin conocimientos previos de programación”. 1ª Edición. 1-3. 2013.

[17] “App Inventor. Características”, disponible en: http://appinventor4.blogspot.com.es/p/blog-page.html [Último acceso: septiembre 2017]

[18] Daum, Berthold. “Profesional Eclipse 3 para desarrolladores Java”. 2-5. 2005

[19] “¿Eclipse o Netbeans?”, disponible en: http://codejavu.blogspot.com.es/2013/10/eclipse-o-netbeans.html [Último acceso: septiembre 2017]

[20] “Desarrollo de juegos con Unity 3D ¿Cómo funciona esta herramienta?”, disponible en: https://www.yeeply.com/blog/desarrollo-de-juegos-con-unity-3d/ [Último acceso: septiembre 2017]

[21] “Presentación del motor de juegos Unity 3D”, disponible en: https://academiaandroid.com/motor-de-juegos-unity-3d/ [Último acceso: septiembre 2017]

[22] “Android Studio VS Eclipse”, disponible en: https://androidstudiofaqs.com/conceptos/android-studio-vs-eclipse [Último acceso: septiembre 2017]

[23] “An introduction to Text-To-Speech in Android”, disponible en: https://android-developers.googleblog.com/2009/09/introduction-to-text-to-speech-in.html [Último acceso: septiembre 2017]

[24] “¿Qué es Google Now y para qué sirve?”, disponible en: http://www.imasdtecnologia.com/imasd-blog/111-aplicaciones/233-que-es-google-now-y-para-que-sirve [Último acceso: septiembre 2017]

[25] “Google Now al detalle: funcionalidades y características”, disponible en: https://www.xatakandroid.com/sistema-operativo/google-now-al-detalle-funcionalidades-y-caracteristicas [Último acceso: septiembre 2017]

[26] “¿Qué es Siri?”, disponible en: http://culturacion.com/que-es-siri/ [Último acceso: septiembre 2017]

[27] “Qué es y cómo funciona Cortana en Windows 10”, disponible en: http://computerhoy.com/noticias/software/que-es-como-funciona-cortana-windows-10-32429 [Último acceso: septiembre 2017]

[28] “Introducción a las Bases de Datos”, disponible en: http://es.ccm.net/contents/66-introduccion-a-las-bases-de-datos [Último acceso: septiembre 2017]

[29] “Qué son las Bases de Datos”, disponible en: http://www.maestrosdelweb.com/que-son-las-bases-de-datos/ [Último acceso: septiembre 2017]

[30] “Introducción a SQLite”, disponible en: http://developeando.net/introduccion-sqlite/ [Último acceso: septiembre 2017]

[31] “Hola SQLite!”, disponible en: http://soyprogramador.liz.mx/hola-sqlite/ [Último acceso: septiembre 2017]

[32] “Conociendo un poco más SQLite”, disponible en: http://soyprogramador.liz.mx/conociendo-un-poco-ms-a-sqlite/ [Último acceso: septiembre 2017]

[33] “android.database.sqlite”, disponible en: https://developer.android.com/reference/android/database/sqlite/package-summary.html [Último acceso: septiembre 2017]

[34] “Android Studio. IDE oficial para Android”, disponible en: https://developer.android.com/studio/index.html [Último acceso: septiembre 2017]

[35] “Survio. Crea encuestas gratis”, disponible en: https://www.survio.com/es/ [Último acceso: septiembre 2017]

[36] “Requisitos legales que debe cumplir una app”, disponible en: http://www.emprendedores.es/gestion/requisitos-legales-app-lanzar-aplicacion [Último acceso: septiembre 2017]

[37] “Subir una aplicación en Google Play”, disponible en: https://support.google.com/googleplay/android-developer/answer/113469?hl=es [Último acceso: septiembre 2017]

[38] “Costos directos e indirectos de un proyecto”, disponible en: http://www.obs-edu.com/es/blog-project-management/viabilidad-de-un-proyecto/costos-directos-e-indirectos-de-un-proyecto [Último acceso: septiembre 2017]

[39] Archivo Excel para cálculo de sueldo descargado del siguiente enlace: http://www.ideasparatuempresa.es/calculadora-calcula-cuanto-te-cuesta-contratar-a-un-nuevo-empleado/ [Último acceso: septiembre 2017]